Out of Sight, Out of Track: Adversarial Attacks on Propagation-based Multi-Object Trackers via Query State Manipulation

Supplementary Material

Supplementary Material Overview

Contents

7. Theories: Modeling TBP Vulnerabilities **2**
7.1. Vuln 1: The Track Query Budget . . . 2
7.2. Vuln 2: The Auto-Regressive Query Update . . . 2

8. Derivation of Temporal Query Flooding **2**
8.1. Component 1: Query Budget Exhaustion . . . 2
8.2. Component 2: Matching Deception . . . 2
8.3. Component 3: Identity Siphoning . . . 3

9. Derivation of Temporal Memory Corruption **3**
9.1. Component 1: Temporal System Weakening . . . 3
9.2. Component 2: State Destruction at the Origin . . . 3

10 Details on Simulated Physical Attacks **4**
10.1 Adversarial Acoustic Injection (AAI) . . . 4
10.1.1 Differentiable Approximation of Blur . . . 4
10.1.2 AAI Motion Blur vs. Gaussian Blur . . . 4
10.1.3 Physical Parameters Calibration and Bounds . . . 5
10.2 Electromagnetic Adversarial Injection (EAI) . . . 5
10.2.1 Differentiable Approximation of Corruption . . . 5
10.2.2 Physical Parameters Calibration and Bounds . . . 6

11 Differentiable Attacks Optimization **6**
11.1 Digital Projected Gradient Descent . . . 6
11.2 Physical Projected Gradient Descent . . . 7
11.2.1 PGD for Acoustic Attack (AAI). . . . 7
11.2.2 PGD for Electromagnetic Attack (EAI) . . . 7

12 Additional Evaluations **8**
12.1 Comparison with Additional Baselines . . . 8
12.2 Ground-Truth Label Independence . . . 8
12.3 TQF Guided Physical Attacks . . . 8
12.3.1 Efficacy of TQF in Physical Constraints . . . 8
12.3.2 TMC vs. TQF in Simulated Physical Settings . . . 9
12.4 Long-Term Memory State Analysis . . . 9
12.4.1 Memory Distinctiveness (Self-Similarity Matrix) . . . 9
12.4.2 Memory Embedding Statistics (Feature Stability) . . . 9
12.4.3 Current-Memory Cross-Similarity . . . 9
12.5 Analysis of Black-box Transferability . . . 10
12.6 FADE under Common Defenses . . . 10
12.6.1 Analysis of Robustness and Defense Efficacy . . . 11
12.7 Practicality, Realism, and Future Work . . . 12

7. Theories: Modeling TBP Vulnerabilities

FADE is a *white-box* attack framework specifically designed to exploit two core architectural assumptions in TBP trackers. We first model these assumptions mathematically to formalize their underlying vulnerabilities (Vuln).

7.1. Vuln 1: The Track Query Budget

At any frame t, a TBP tracker is given a candidate set of $M+N$ queries, $\mathcal{Q}_t^{in} = \mathcal{T}_{t-1} \cup \mathcal{Q}_{\text{det}}$. However, the tracker has a fixed, finite query budget B for the *next* frame's track queries, $\mathcal{T}_t$, where $|\mathcal{T}_t| \leq B$ and typically $M+N > B$.

The tracker must implicitly solve a constrained resource allocation problem: it must select the B *best* queries to propagate. We can model this as an optimization problem where the tracker seeks to maximize the total *utility* (e.g., confidence or objectness score) of the propagated set:

$$\max_{\mathcal{T}_t \subseteq \mathcal{Q}_t^{out}} \sum_{q_i \in \mathcal{T}_t} s(q_i) \quad \text{subject to} \quad |\mathcal{T}_t| \leq B \tag{9}$$

where $s(q_i)$ is the confidence score for the output query q_i.

Attack Rationale ($\mathcal{L}_{\text{TQF}}$). The Temporal Query Flooding (TQF) attack exploits this zero-sum constraint. We introduce a set of K adversarial queries $\mathcal{T}_{\text{adv}}$ and use an optimization objective $\mathcal{L}_{\text{TQF}}$ to force $s(q_j) \to 1$ for all $q_j \in \mathcal{T}_{\text{adv}}$. If the attacker can make its K adversarial queries appear to have a higher utility $s(q_j)$ than K legitimate track queries $q_i \in \mathcal{T}_t$, the tracker's optimization logic will be forced to discard the legitimate tracks to make room for the adversarial ones, hence the *flooding* that leads to query starvation.

7.2. Vuln 2: The Auto-Regressive Query Update

The core of TBP tracking is its auto-regressive nature. We can model the temporal propagation of a single track's identity as a *discrete-time dynamical system*.

Let $\mathbf{h}_i^t \in \mathbb{R}^D$ be the hidden state (the D-dimensional query embedding) for object i at frame t. The Query Updater $\mathcal{F}$ acts as the state transition function, where q_i^{out} is the output query from the decoder and X_t is the image feature map:

$$(\mathbf{h}_i^t, \mathcal{M}_t) = \mathcal{F}(\mathbf{h}_i^{t-1}, \mathcal{M}_{t-1}, q_i^{out}, X_t) \tag{10}$$

A *stable* track requires that the identity of the object is preserved across time steps. Mathematically, this implies that the hidden state $\mathbf{h}_i^t$ must remain *close* to its predecessor $\mathbf{h}_i^{t-1}$ in the embedding manifold, representing the same object. We formalize the stability condition as follows:

$$\textit{Stability Condition:} \quad \text{sim}(\mathbf{h}_i^t, \mathbf{h}_i^{t-1}) \geq \tau_{\text{sim}} \tag{11}$$

where $\text{sim}(\cdot,\cdot)$ is a similarity metric (e.g., cosine similarity) and τ_{sim} is a high threshold (e.g., ≈ 1.0).

Attack Rationale ($\mathcal{L}_{\text{TMC}}$). The Temporal Memory Corruption (TMC) attack is a direct *destabilization attack* on the dynamical system of the auto-regressive query update logic. Instead of adding new queries, it seeks to violate the stability condition for existing, legitimate tracks.

1. $\mathcal{L}_{\text{Decorr}}$ is designed to *directly minimize* $\text{sim}(\mathbf{h}_i^t, \mathbf{h}_i^{t-1})$, forcing a temporal disconnect.
2. $\mathcal{L}_{\text{Erase}}$ is a more powerful destructive attack, seeking to send the state to a null point, $\mathbf{h}_i^t \to \mathbf{0}$, from which no identity can be recovered.

8. Derivation of Temporal Query Flooding

$\mathcal{L}_{\text{TQF}}$ is a composite loss designed to solve the query budget exhaustion problem from Sec. 7.1.

$$\mathcal{L}_{\text{TQF}} = \lambda_{\text{Flood}}\mathcal{L}_{\text{Flood}} + \lambda_c\mathcal{L}_{\text{Cost}} + \lambda_s\mathcal{L}_{\text{Siphon}}$$

8.1. Component 1: Query Budget Exhaustion

Objective. Force the tracker to assign maximal confidence to adversarial queries.

$$\mathcal{L}_{\text{Flood}} = \mathbb{E}_{q_i \in \mathcal{T}_{\text{adv}}}\left[(1.0 - \max_c \sigma(z_i)_c)^2\right]$$

Justification. Let $\hat{c}_i = \max_c \sigma(z_i)_c$ be the tracker's predicted confidence score, $s(q_i)$, for query q_i. This loss function formulates the attack as a regression problem, minimizing the L2 (squared Euclidean) distance between the current confidence $\hat{c}_i$ and a target value of 1.0.

While a cross-entropy loss could be used, the squared L2 loss is a *hard-margin* objective that provides a strong gradient for any $\hat{c}_i < 1.0$. It aggressively punishes all adversarial queries that are not-perfectly-confident, ensuring the entire set $\mathcal{T}_{\text{adv}}$ presents itself as a block of high-utility *distractors*, thereby maximizing its ability to displace legitimate tracks in the constrained allocation problem (Eq. 9).

8.2. Component 2: Matching Deception

Objective. Deceive the bipartite matching algorithm (e.g., hangarian matching) by making adversarial queries appear to be a low-cost match for real objects.

$$\mathcal{L}_{\text{Cost}} = -\mathbb{E}_{g_j \in \mathcal{G}_t}\left[\log \sum_{q_i \in \mathcal{T}_{\text{adv}}} \exp(-C(q_i, g_j))\right]$$

Justification. This formulation is a direct application of the LogSumExp (LSE) method to create a differentiable approximation of the $\min$ function. First, recall the LSE function, which is a smooth approximation of the $\max$ function:

$$\text{LSE}(\mathbf{x}) = \log \sum_i \exp(x_i) \approx \max(\mathbf{x})$$

We can derive a smooth min function by noting that $\min(\mathbf{x}) = -\max(-\mathbf{x})$.

$$\min(\mathbf{x}) = -\max(-\mathbf{x}) \tag{12}$$
$$\approx -\text{LSE}(-\mathbf{x}) \tag{13}$$
$$= -\log \sum_i \exp(-x_i) \tag{14}$$

Now, consider our $\mathcal{L}_{\text{Cost}}$ for a single ground-truth object g_j. Let $\mathbf{C}_j = \{C(q_i, g_j) | q_i \in \mathcal{T}_{\text{adv}}\}$ be the vector of matching costs between g_j and all adversarial queries.

$$\mathcal{L}_{\text{Cost}}(g_j) = -\log \sum_{q_i \in \mathcal{T}_{\text{adv}}} \exp(-C(q_i, g_j)) = -\text{LSE}(-\mathbf{C}_j)$$

Therefore, $\mathcal{L}_{\text{Cost}}(g_j)$ is a differentiable approximation of $\min(\mathbf{C}_j)$.

$$\mathcal{L}_{\text{Cost}}(g_j) \approx \min_{q_i \in \mathcal{T}_{\text{adv}}} C(q_i, g_j)$$

By minimizing $\mathcal{L}_{\text{Cost}}$, our attack is *minimizing the minimum matching cost*. This creates an adversarial query q_i that appears to be an extremely reliable (low-cost) match for a real object g_j, fooling the tracker's association logic.

8.3. Component 3: Identity Siphoning

Objective. Make new adversarial queries appear to be the continuation of old, reliable, legitimate tracks.

$$\mathcal{L}_{\text{Siphon}} = -\mathbb{E}_{h_i^t \in \mathcal{H}_{\text{adv}}^t, h_j^{t-1} \in \mathcal{H}_{\text{anchor}}^{t-1}} \left[\text{cosine}(\mathbf{h}_i^t, \mathbf{h}_j^{t-1})\right]$$

Justification. This loss is equivalent to *maximizing* the cosine similarity between a new adversarial query state $\mathbf{h}_i^t$ and a historical, legitimate track state $\mathbf{h}_j^{t-1}$:

$$\max_{\boldsymbol{\omega}} \mathbb{E}\left[\frac{\mathbf{h}_i^t \cdot \mathbf{h}_j^{t-1}}{\|\mathbf{h}_i^t\|_2 \|\mathbf{h}_j^{t-1}\|_2}\right]$$

Geometric Interpretation (Cosine vs. L2 Distance). Here we justify the choice of cosine similarity over L2 distance. In TBP trackers, the *identity* of an object is encoded in the *direction* of its D-dimensional query vector $\mathbf{h}$. The *magnitude* $\|\mathbf{h}\|$ often encodes non-identity features like confidence or visibility.

- An L2 loss, $\mathcal{L}_{\text{L2}} = \|\mathbf{h}_i^t - \mathbf{h}_j^{t-1}\|_2^2$, would penalize differences in both direction and magnitude.
- Our chosen cosine loss, $\mathcal{L}_{\text{Siphon}}$, *only* penalizes differences in direction.

By maximizing the cosine similarity, we are minimizing the angle θ between the two vectors, forcing $\mathbf{h}_i^t$ to align geometrically with $\mathbf{h}_j^{t-1}$ in the embedding manifold. The Query Updater $\mathcal{F}$ (from Sec. 7.2) interprets this directional alignment as a re-emergence of the *same object*, thus *siphoning* the legitimate track's identity.

9. Derivation of Temporal Memory Corruption

$\mathcal{L}_{\text{TMC}}$ is the *destabilization* attack from Sec. 7.2.

$$\mathcal{L}_{\text{TMC}} = \lambda_{\text{Decorr}} \mathcal{L}_{\text{Decorr}} + \lambda_{\text{Erase}} \mathcal{L}_{\text{Erase}}$$

9.1. Component 1: Temporal System Weakening

Objective. Sever the temporal link by violating the track stability condition in Eq. 11.

$$\mathcal{L}_{\text{Decorr}} = \mathbb{E}_{i \in \text{matched}} \left[\text{cosine}(\mathbf{h}_i^t, \mathbf{h}_i^{t-1})\right]$$

Justification. As established in our dynamical system model (Eq. 10), track stability requires $\text{cosine}(\mathbf{h}_i^t, \mathbf{h}_i^{t-1}) \approx 1$. The loss $\mathcal{L}_{\text{Decorr}}$ is a direct objective to *minimize* this value. In non-negative feature spaces (common after ReLU nonlinearities), the effective minimum is 0.

$$\min_{\boldsymbol{\omega}} \mathbb{E}_{i \in \text{matched}} \left[\frac{\mathbf{h}_i^t \cdot \mathbf{h}_i^{t-1}}{\|\mathbf{a}\|_2 \|\mathbf{b}\|_2}\right] \quad \Longrightarrow \quad \mathbf{h}_i^t \cdot \mathbf{h}_i^{t-1} \to 0$$

Minimizing this loss forces the current state vector $\mathbf{h}_i^t$ to become *orthogonal* to its predecessor $\mathbf{h}_i^{t-1}$ (i.e., $\mathbf{h}_i^t \perp \mathbf{h}_i^{t-1}$). Geometrically, two orthogonal vectors share no directional information. This represents a break in the state's temporal linkage, guaranteeing a violation of the stability condition and causing the Query Updater to fail re-association.

9.2. Component 2: State Destruction at the Origin

Objective. Destroy the track's identity by collapsing its feature embedding.

$$\mathcal{L}_{\text{Erase}} = \mathbb{E}_{h_i \in \mathcal{H}_{\text{matched}}^t} \left[\|\mathbf{h}_i\|_2^2\right]$$

Justification. This loss minimizes the squared L2-norm (the *energy*) of the feature vector $\mathbf{h}_i^t$.

- **Information-Theoretic View:** The origin vector $\mathbf{0}$ is an information-theoretic *null* state. It carries no information about identity, appearance, or motion. Minimizing this loss is an *erasure* attack that drives the information content of the feature vector to zero.
- **Geometric View:** The unique global minimum of this convex loss function is $\mathbf{h}_i^t = \mathbf{0}$, the origin of the D-dimensional embedding space.

Forcing the state to collapse to the origin is mathematically destructive for the tracker, as all similarity metrics fail:

1. **Dot Product Similarity:** $\text{dot}(\mathbf{h}_i^t, \mathbf{v}) = \text{dot}(\mathbf{0}, \mathbf{v}) = 0$ for any vector $\mathbf{v}$.
2. **Cosine Similarity:** $\text{cosine}(\mathbf{h}_i^t, \mathbf{v}) = \frac{\mathbf{0} \cdot \mathbf{v}}{\|\mathbf{0}\|_2 \|\mathbf{v}\|_2}$, which is *undefined* due to division by zero.

This attack does not merely *confuse* the tracker; it breaks the underlying mathematics of the similarity metrics it relies upon. It erases the track state, rendering future re-association impossible.

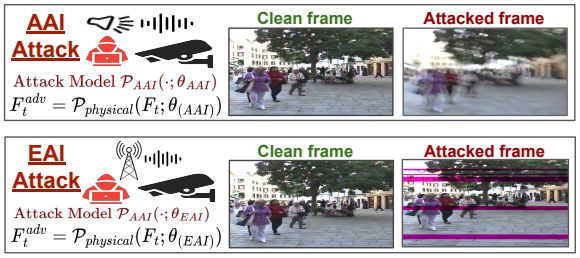

Figure 5. Physical Adversarial Attacks Scenarios and Models.

10. Details on Simulated Physical Attacks

The following provides details on the simulated physical attack models, specifically covering their parameters, and visualizations showing how those parameters vary. We illustrate the attack scenarios and threat models in Fig.5

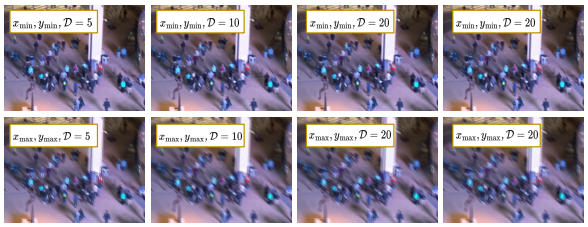

Figure 6. Illustration of the adversarial frame F_t^{adv} from the $\mathcal{P}_{AAI}$ transformation. The top row when fixing x, y, ϕ at minimum values while varying hyperparameter $\mathcal{D}$ from 5 (low) to 30 (high). The top row when fixing x, y, ϕ at maximum values while varying hyperparameter $\mathcal{D}$ from 5 (low) to 30 (high).

10.1. Adversarial Acoustic Injection (AAI)

The AAI attack simulates the physical effect of sound waves on a camera sensor's mechanical stabilization systems. In modern cameras, Micro-Electro-Mechanical Systems (MEMS) stabilizers are highly susceptible to vibrations induced by carefully tuned acoustic signals, particularly within the ultrasonic frequency range [10, 41]. This physical vibration causes a subtle, high-frequency motion of the camera sensor during the frame's exposure time, which visually manifests as motion blur in the captured image and can be exploited for adversarial intents [10, 41]

To accurately simulate this physically-plausible phenomena in a differentiable manner, we model the camera's motion path as a continuous, sinusoidal oscillation. Our AAI simulation, represented by the function $\mathcal{P}_{AAI}(\cdot;\theta_{AAI})$, does not rely on a conventional convolution with a blur kernel. Instead, it utilizes a spatial transformation-based approach that directly models the effect of sensor movement and aggregates the resulting visual data. The perturbed frame F_t^{adv} is generated from a clean frame F_t according to the following equation:

$$F_t^{adv} = \mathcal{P}_{AAI}(F_t;\theta_{AAI})$$

where the attack's learnable physical parameters are defined as $\theta_{AAI} = (x, y, \phi)$. The parameters x and y represent the maximum amplitude of the horizontal and vertical oscillations, respectively, while ϕ is a phase offset for the sinusoidal motion.

10.1.1. Differentiable Approximation of Blur

In a physical AAI attack, the MEMS sensor resonance ($f_{\text{res}} \approx 26$kHz) is significantly higher than the camera framerate (30fps). This results in approximately $N_{cyc} \approx 866$ oscillation cycles per exposure duration T_{exp}.

Simulating all 866 cycles in a differentiable loop is computationally expensive. However, we observe that the *Point Spread Function (PSF)*, the statistical distribution of the pixel displacement, converges rapidly. For a sinusoidal displacement $d(t) = A\sin(\omega t)$, the intensity distribution of the blur kernel follows a bounded *U-shaped* distribution (the Arcsine distribution), as shown in Fig.7, regardless of the frequency ω, provided $\omega \gg 1/T_{\text{exp}}$. Therefore, we approximate the high-frequency integral using a *single-period Monte Carlo approximation*. We model the blur kernel by sampling the displacement trajectory $d(t)$ at $\mathcal{D}$ discrete time steps evenly spaced over one phase cycle $[-\pi, \pi]$:

$$F_t^{\text{adv}} \approx \frac{1}{\mathcal{D}} \sum_{k=1}^{\mathcal{D}} \mathcal{T}r(F_t, \boldsymbol{\delta}_k) \quad \text{where} \quad \boldsymbol{\delta}_k = (x, y) \cdot \sin\left(\frac{2\pi k}{\mathcal{D}}\right)$$

where $\mathcal{T}r(\cdot)$ is the differentiable affine transformation operation (implemented via Spatial Transformer Networks). The full simulation process unfolds as follows:

1. A set of discrete offsets, $\{\boldsymbol{\delta}_k\}_{k=1}^{\mathcal{D}}$, is generated based on the sinusoidal function controlled by parameters x, y, ϕ.
2. For each step k, a corresponding transformation grid G_k is created. This grid encodes the spatial shift to be applied to the entire image.
3. The original image F_t is spatially transformed at each step k using its respective grid G_k, resulting in a set of $\mathcal{D}$ perturbed image instances, $\{S_k\}_{k=1}^{\mathcal{D}}$.
4. The final motion-blurred frame F_t^{adv} is obtained by averaging these transformed instances:

$$F_t^{adv} = \frac{1}{\mathcal{D}} \sum_{k=1}^{\mathcal{D}} S_k$$

In our experiments, we found that $\mathcal{D} = 10$ samples provides a sufficient approximation of the Optical Image Stabilization (OIS) blur kernel while maintaining high efficiency. This pipeline is fully differentiable, allowing the adversarial loss to back-propagate through the physical simulation and directly optimize θ_{AAI}.

10.1.2. AAI Motion Blur vs. Gaussian Blur

As illustrated in Fig. 7, the simulated blur kernel differs significantly from standard Gaussian or defocus blur. Because the AAI attack induces a harmonic vibration $d(t) =$

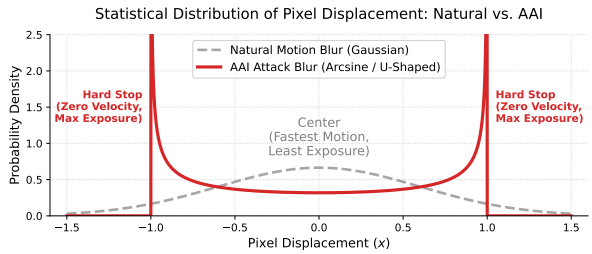

Figure 7. Natural Gaussian blur motion Vs. AAI-induced blur.

$\alpha \sin(\omega t)$, the camera sensor spends the majority of the exposure time near the physical limits of the oscillation (the peaks and troughs), where the instantaneous velocity of the sensor approaches zero ($v(t) \to 0$). Mathematically, the probability density function of this displacement follows an Arcsine distribution:

$$p(x) = \frac{1}{\pi\sqrt{\alpha^2 - x^2}}, \quad x \in (-\alpha, \alpha)$$

which approaches infinity at the boundaries $x \to \pm\alpha$. Visually, this does not merely *smear* the object; it creates a distinct *double-edge* or *ghosting* effect.

10.1.3. Physical Parameters Calibration and Bounds

To ensure our FADE-AAI attack remains physically realizable, we constrain our learnable parameters based on the hardware limitations and experimental calibration data established in the Poltergeist framework [10].

***(i)* Resonance Frequency Bounds.** The effectiveness of acoustic injection is limited to the resonant bandwidth of the MEMS sensor in the targeted camera. Outside this narrow band, the induced drift is negligible.

- **Calibration Method:** The resonant frequency is identified via a frequency sweep (0Hz–30kHz) while monitoring the MEMS sensor raw output.
- **Target Hardware:** For example, for the Samsung Galaxy S20 (SM-G981U) utilized in our AAI simulation model, the STMicroelectronics LSM6DSO MEMS sensor exhibits a sharp resonance peak at $f_{\text{res}} \approx 26.3$kHz.
- **Constraint:** In our simulation, f_{res} is treated as a fixed environmental constant, set to 26.3kHz.

***(ii)* Resonance Amplitude Bounds.** The learnable displacement parameters (x, y) correspond to the amplitude of the induced oscillatory drift. This amplitude is physically constrained by the maximum Sound Pressure Level (SPL) and the device's acoustic transfer function. We bound the magnitude $\|\boldsymbol{\alpha_m}\|$ using the empirical log-linear relationship derived in [10]:

$$\alpha_{\text{drift}} \propto 10^{\frac{\text{SPL}_{dB}}{20}}$$

- **Constraint:** We clamp $\boldsymbol{\alpha_m}$ such that the induced angular velocity does not exceed ± 2.0rad/s. This corresponds to an acoustic injection of approx. 110dB at the sensor, the upper limit of standard ultrasonic arrays before non-linear distortion dominates.

***(iii)* OIS Physical Saturation Limits ($D_{\max}$).** The Optical Image Stabilization (OIS) system compensates for drift by physically shifting the lens. This movement is bounded by the Voice Coil Motor (VCM) travel range.

- **Calibration:** Experimental results show that OIS systems typically saturate at displacements equivalent to $1\% \sim 3\%$ of the image width.
- **Hard Constraint:** We apply a differentiable clamping function to the computed displacement trajectory $d(t)$:

$$\|d(t)\|_2 \leq D_{\max}$$

where $D_{\max} = 0.03 \times W$ (approx. 50 pixels for a 1080p stream). Any optimization attempting to push the lens beyond this limit yields diminishing returns, simulating the physical *hard stop* of the OIS mechanism.

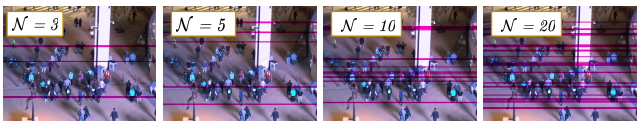

Figure 8. Illustration of the adversarial frame F_t^{adv} from the $\mathcal{P}_{EAI}$ transformation. The top row when applying random color stripes while varying $\mathcal{N}$ from 3 (low) to 20 (high).

10.2. Electromagnetic Adversarial Injection (EAI)

The EAI attack simulates the effects of electromagnetic interference (EMI) that directly corrupts a camera's electronic signal transmission. A primary target is the Mobile Industry Processor Interface (MIPI) CSI-2 lanes or the Analog-to-Digital Converter (ADC) path, responsible for transmitting the raw sensor data. Such corruption introduces structured noise patterns and color artifacts into the raw image data, which are then propagated through the camera's internal Image Signal Processor (ISP) pipeline [12, 22, 39].

To simulate this physically-plausible phenomenon in a differentiable manner, we introduce a simulator function $\mathcal{P}_{EAI}(\cdot; \theta_{EAI})$ that mimics the stages of a camera's image acquisition and processing. The perturbed frame F_t^{adv} is generated as:

$$F_t^{adv} = \mathcal{P}_{EAI}(F_t; \theta_{EAI})$$

where the attack's learnable physical parameters are defined as $\theta_{EAI} = ((\mathbf{r}, \mathbf{w}) \in \mathbb{M})$. Here, $\mathbf{r} \in \mathbb{R}^{\mathbb{N}}$ represents the vertical row indices (timing delays) of the interference, and $\mathbf{w} \in \mathbb{R}^{\mathbb{N}}$ represents the width (duration) of each corrupted stripe. The number of stripes, $\mathbb{N}$, is a fixed hyperparameter representing the duty cycle limit of the EMI injector.

10.2.1. Differentiable Approximation of Corruption

Unlike standard pixel-noise attacks, EAI corruption occurs in the raw signal domain *before* image processing. We

model this by inverting the ISP pipeline. Our models is based on the GlitchHiker pipeline and implementation [12]. The simulation process unfolds as follows:

***(i)* Raw Sensor Simulation (Inverse ISP):** The clean input image F_t is first converted into a simulated RGGB Bayer pattern F_{Bayer} using a differentiable masking function that mimics the initial sensor readout.

$$F_{\text{Bayer}} = F_t \odot \mathbb{M}_{\text{CFA}}$$

***(ii)* Sensor-Level Corruption:** We simulate the *Glitch* attack mode identified in [12] by selectively zeroing out specific color channels within the Bayer pattern. This mimics the suppression of the differential voltage signal during the EMI pulse.

$$F_{\text{Corrupted\_Raw}} = F_{\text{Bayer}} \odot \mathbb{M}_{\text{drop}}$$

where $\mathbb{M}_{\text{drop}}$ here zeros out the Green channel (the luminance carrier), which is a characteristic signature of MIPI synchronization failures.

***(iii)* Demosaicing Reconstruction:** The corrupted Bayer pattern is reconstructed into a full RGB image via a differentiable bilinear demosaicing algorithm $\mathcal{D}_{demosaic}$. This step generates the non-linear *zippering* and false-color artifacts (e.g., purple stripes) that are distinct from additive noise.

$$F_{\text{Artifact}} = \mathcal{D}_{demosaic}(F_{\text{Corrupted\_Raw}})$$

***(iv)* Differentiable Stripe Masking:** To optimize the location of the glitches, a differentiable soft mask $\mathbb{M}_{\text{Soft}}$ is constructed from the learnable parameters $\mathbf{r}$ and $\mathbf{w}$ using a sigmoid product function:

$$\mathbb{M}_{\text{Soft}}(y) = \max_{k=1}^{N} [\sigma(s(y - r_k)) \cdot \sigma(s(r_k + w_k - y))]$$

This mask allows gradients to flow into the row coordinates, enabling the attacker to learn *where* to inject the pulse. The final adversarial frame is a blend:

$$F_t^{adv} = F_t \odot (1 - \mathbb{M}_{\text{Soft}}) + F_{\text{Artifact}} \odot \mathbb{M}_{\text{Soft}} \odot \lambda$$

where λ scales the intensity of the corruption.

This entire pipeline is fully differentiable, allowing the adversarial loss to back-propagate through the demosaicing algorithm and directly optimize θ_{EAI}.

10.2.2. Physical Parameters Calibration and Bounds

To ensure the optimized parameters correspond to a realizable EMI attack, we constrain θ_{EAI} based on the hardware limitations of standard EMI injection devices (e.g., spark-gap generators or amplified SDRs).

***(i)* Pulse Width Constraints (w).** The width of the glitch w_k corresponds to the duration of the EMI pulse. Hardware capacitors limit the minimum and maximum pulse duration.

- **Constraint:** $w_{\min} \leq w_k \leq w_{\max}$. We set $w_{\min} \approx 5$ rows (approx. $100\mu s$) and $w_{\max} \approx 50$ rows, reflecting the discharge limits of portable EMI injectors.

***(ii)* Pulse Frequency / Count ($\mathbb{N}$).** High-voltage EMI sources have a recharge delay (duty cycle limit). An attacker cannot jam every single row in a frame.

- **Constraint:** We fix the maximum number of glitches $\mathbb{N}$ (e.g., $\mathbb{N} = 20$) to model a realistic recharge cycle, forcing the optimizer to select only the most impactful temporal windows (image rows) for injection.

***(iii)* VSYNC Synchronization (r).** The row index r_k is physically controlled by the time delay Δt of the pulse injection relative to the camera's VSYNC interrupt signal.

- **Constraint:** $0 \leq r_k \leq H_{\text{image}}$. The position is fully continuous within the frame readout time, allowing precise targeting of specific object features.

11. Differentiable Attacks Optimization

11.1. Digital Projected Gradient Descent

We detail the digital PGD attack. This method generates pixel-level adversarial perturbations by directly optimizing the pixel values of the input image. Unlike physical attacks, which are constrained by the plausibility of their underlying parameters, digital attacks are constrained by the magnitude of the perturbation in the pixel space. The attack is implemented using a PGD-style iterative optimization loop, which aims to maximize a specific attack loss by taking small, controlled steps in the direction of the loss gradient.

Configuration. The PGD attacker is configured with the following key parameters:

- Maximum Perturbation (ϵ): Defines the maximum allowed magnitude of the total perturbation in the pixel space. The attack generates perturbations such that they are constrained to an L_∞ norm ball, i.e., $||\delta||_\infty \leq \epsilon$. We set ϵ to 8/255.
- Step Size (α): The size of each step taken during the iterative optimization. It determines how quickly the perturbation is updated. A small α ensures small, controlled updates. We set α to 1/255.
- Number of Steps ($\mathcal{K}$): The number of iterations for the optimization loop. The attack runs for $\mathcal{K} = 50$ steps.

Mechanism. The digital PGD attack is an iterative process that calculates the gradient of the attack loss with respect to the input image and updates the perturbation accordingly. The process, outlined in Algorithm 1, begins by initializing the perturbation δ to a small value (often zero). In each iteration, a temporary adversarial image $F_t^{adv} = F_t + \delta$ is generated. Then, a forward pass of this perturbed image is performed on the tracker f_{MOT}, and the attack loss $\mathcal{L}_{adv}$ is computed. The gradient $\nabla_\delta \mathcal{L}_{\text{adv}}$ is back-propagated to determine the direction of the highest loss increase. The perturbation δ is then updated by taking a step in this direc-

tion, and is subsequently clipped to ensure it remains within the predefined L_∞ constraint (ϵ). The final perturbed image is then produced after $\mathcal{K}$ iterations. In our implementation we freeze the model's weights, ensuring that gradients are computed only with respect to the input perturbation, and preventing any changes to the model's parameters.

Algorithm 1 Digital PGD Attack Algorithm

Input: MOT Tracker f_{MOT}, Frame F_t, Attack Parameters (Perturbation size: ϵ, Step size: α, Iterations: T)
Output: F_t^{adv}

Initialize perturbation $\delta \leftarrow \mathbf{0}$
for k = 1 to T **do**
  $F_t^{adv} \leftarrow \text{Clamp}(F_t + \delta, \text{min}, \text{max})$
  Get loss $\mathcal{L}_{\text{adv}}(f_{\text{MOT}}(F_t^{adv}(\omega)), \mathcal{G}_t)$
  Compute gradient $\nabla_\delta \mathcal{L}_{\text{adv}}$
  $\delta \leftarrow \delta + \alpha \cdot \text{sign}(\nabla_\delta \mathcal{L}_{\text{adv}})$
  $\delta \leftarrow \text{Clamp}(\delta, -\epsilon, \epsilon)$
end for
return $F_t^{adv} \leftarrow \text{Clamp}(F_t + \delta, \text{min}, \text{max})$

11.2. Physical Projected Gradient Descent

The key distinction from the digital attack is that PGD is applied to the physical parameters (θ) rather than the pixel values (δ). Unlike the digital PGD attacker, which directly optimizes perturbations in the pixel space, the physical PGD attacker optimizes the parameters of a differentiable physical simulation model. The core principle remains the same: iteratively taking steps in the direction of the loss gradient to maximize an attack objective $\mathcal{L}_{\text{adv}}$. However, in this case, the gradient is computed with respect to the physical parameters $\theta_{(AAI||EAI)}$, and the projection step ensures these parameters remain within their plausible physical ranges, rather than constraining the pixel-space perturbation.

11.2.1. PGD for Acoustic Attack (AAI).

The PGD for the AAI attack optimizes the parameters of the motion blur simulation. The algorithm iteratively refines the physical parameters of the simulated camera motion to maximize the adversarial loss $\mathcal{L}_{\text{TMC}}$.

Configuration. The PGD loop for AAI operates on the physical parameters $\theta_{AAI} = (x, y, \phi)$, which represent the amplitudes of horizontal and vertical sinusoidal oscillations, and a phase offset, respectively. These parameters are bounded by a predefined plausible range ($[min_x, max_x]$, $[min_y, max_y]$, $[min_\phi, max_\phi]$). The number of blur samples, $\mathcal{D}$, is a fixed hyperparameter for a given attack configuration. The step size, α, is applied to each physical parameter independently during optimization. The attack runs for $\mathcal{K} = 150$ steps.

Mechanism: The PGD for AAI, outlined in Algorithm 2, initializes the parameters x, y, ϕ to their minimum values. In each iteration, a perturbed image F_t^{adv} is generated by passing the current image and the current parameters to the differentiable AAI simulation model $\mathcal{P}_{AAI}$. A forward pass on the target tracker f_{MOT} is performed with F_t^{adv}, and the adversarial loss $\mathcal{L}_{TMC}$ is computed. The gradients of this loss with respect to x, y, ϕ are then calculated. The parameters are updated using a signed gradient step and are subsequently clamped back into their predefined physical ranges, which serves as the projection step in this optimization.

Algorithm 2 PGD for Acoustic Attack (FADE-AAI)

Input: Tracker f_{MOT}, Frame F_t, AAI Attack Parameters ($\theta_{AAI} = (x, y, \phi, \mathbb{D})$, Step size: α, Iterations: T)
Output: F_t^{adv}

Initialize physical parameters $x, y, \phi \in \theta_{AAI}$
for k = 1 to $\mathcal{K}$ **do**
  $F_t^{adv} \leftarrow \mathcal{P}_{AAI}(F_t; x, y, \phi)$
  Get loss $\mathcal{L}_{\text{adv}}(f_{\text{MOT}}(F_t^{adv}(\omega)), \mathcal{G}_t)$
  Compute gradients $\nabla_x, \nabla_y, \nabla_\phi \mathcal{L}_{\text{adv}}$
  $x \leftarrow x + \alpha \cdot \text{sign}(\nabla_x)$
  $y \leftarrow y + \alpha \cdot \text{sign}(\nabla_y)$
  $\phi \leftarrow \phi + \alpha \cdot \text{sign}(\nabla_\phi)$
  $\text{Clamp}_x(x), \text{Clamp}_y(y), \text{Clamp}_\phi(\phi)$
end for
return $F_t^{adv} \leftarrow \mathcal{P}_{AAI}(F_t; x, y, \phi)$

11.2.2. PGD for Electromagnetic Attack (EAI)

The PGD for the EAI attack optimizes the spatial parameters that define the adversarial stripes. The algorithm iteratively refines the location and dimensions of these corrupted regions to maximize the adversarial loss $\mathcal{L}_{TMC}$.

Configuration. PGD for EAI operates on the $\mathbb{M} \in \theta_{EAI}$, which is a 2D matrix where each row $[r, w]$ defines a horizontal stripe's row ID and width. The maximum number of stripes, $\mathbb{N}$, is a fixed hyperparameter for a given attack. The step size, α, is applied to all parameters within $\mathbb{M}$ simultaneously. The plausible range for these parameters (e.g., r within image height, w within valid bounds) is enforced as a projection constraint. The attack runs for $\mathcal{K} = 150$.

Mechanism. The PGD algorithm for EAI, outlined in Algorithm 3, initializes the $\mathbb{M}$ to a predefined configuration (e.g., uniform, random). In each iteration, a perturbed image F_t^{adv} is generated by passing the current $\mathbb{M}$ to the differentiable EAI simulation model $\mathcal{P}_{EAI}$. A forward pass on the target tracker f_{MOT} is performed with F_t^{adv}, and the adversarial loss $\mathcal{L}_{TMC}$ is computed. The gradients of this loss with respect to $\mathbb{M}$ are then calculated. The $\mathbb{M}$ tensor is updated using a signed gradient step. The projection back into the plausible parameter space is implicitly handled by clamping functions within the EAI simulation model.

Algorithm 3 PGD for Electromagnetic Attack (EAI)

Input: Tracker f_{MOT}, Frame F_t, AAI Attack Parameters $(\theta_{EAI} = (\mathbb{M}, \mathbb{N})$, Step size: α, Iterations: $T)$
Output: F_t^{adv}

Initialize $\mathbb{M}, \theta_{EAI} \in \theta_{EAI}$
for k = 1 to $\mathcal{K}$ **do**
  $F_t^{adv} \leftarrow \mathcal{P}_{EAI}(F_t; \theta_{EAI})$
  Get loss $\mathcal{L}_{attack}(\mathcal{M}(F_t^{adv}), \mathcal{G}_t)$
  Compute gradient $\nabla_{\theta_{EAI}} \mathcal{L}_{attack}$
  $\theta_{EAI} \leftarrow \theta_{EAI} + \alpha \cdot \text{sign}(\nabla_{\theta_{EAI}} \mathcal{L}_{attack})$
  $\theta_{EAI} \leftarrow \text{Clip}_{\Theta}(\theta_{EAI})$
end for
return $F_t^{adv} \leftarrow \mathcal{P}_{EAI}(F_t; \theta_{EAI})$

12. Additional Evaluations

We provide additional evaluations of FADE, including further comparison with baselines, ablation studies, transferability analysis, results of the TQF-guided simulated physical attacks, analysis of the long-term memory integrity in TBP trackers, and the efficacy of FADE against defenses.

Table 7. FADE vs. Untargeted PGD and recent MOT attack.

Attack	Dataset	MOTR	MOTRv2	MeMOTR	Samba	CO-MOT
Clean	MOT17 | MOT20	58.63 | 55.40	59.96 | 56.70	67.35 | 68.20	62.91 | 64.50	58.16 | 61.20
Untargeted PGD	MOT17 | MOT20	49.90 | 40.10	57.41 | 57.77	59.85 | 60.74	58.38 | 57.61	55.85 | 60.95
BBA_Blind	MOT17 | MOT20	48.51 | 41.19	56.07 | 59.36	59.10 | 58.47	56.79 | 54.27	53.02 | 57.56
BBA_Blur	MOT17 | MOT20	46.92 | 35.85	57.13 | 58.13	62.54 | 64.67	56.52 | 51.63	53.55 | 58.97
FADE$_{\text{TMC}}$	MOT17 | MOT20	45.90 | 42.63	39.29 | 29.64	41.41 | 57.67	45.53 | 46.85	37.26 | 33.37
FADE$_{\text{TQF}}$	MOT17 | MOT20	45.89 | 40.38	46.76 | 56.68	41.56 | 37.70	48.04 | 54.91	41.73 | 49.28

12.1. Comparison with Additional Baselines

As FADE represents the first adversarial framework specifically targeting TBP trackers, native baselines designed for this architecture are currently absent in the literature. For a rigorous evaluation, we extend the cross-paradigm comparisons presented in the main paper by evaluating FADE against three additional baseline strategies: standard *Untargeted PGD*, and two spatial-degradation attacks, *BBA-Blind* and *BBA-Blur* [20]. Several key insights emerge from this extended comparison. While spatial-degradation attacks can degrade performance in high-density scenes or on weaker detection backbones, they fail to induce the systemic identity collapse observed with FADE, which achieves disproportionately higher degradation on robust, memory-augmented models such as MeMOTR, Samba, and CO-MOT. Furthermore, the significant performance gap between FADE and Untargeted PGD verifies that TBP vulnerabilities are fundamentally architectural; standard gradient-based noise is insufficient to disrupt the recurrent state propagation that FADE specifically poisons. Ultimately, while standard MOT attacks are primarily designed for spatial disruption, FADE exploits the temporal dependencies of the query-update logic to trigger a non-transient *forgetting* state that standard baselines cannot replicate.

12.2. Ground-Truth Label Independence

To verify the practical feasibility of the TQF attack, we investigate the impact of replacing privileged Ground Truth (GT) information with online tracker predictions for the cost mimicry loss ($\mathcal{L}_{\text{Cost}}$). While the primary experiments in Section 4.7.1 utilize GT to set a performance upper-bound for the attack, the results summarized in Table 8 confirm that FADE operates with near-identical efficacy using surrogate labels derived directly from the tracker's own predictions. We observe a negligible average HOTA delta of ≤ 0.2 points across all tested models when transitioning from GT to predicted labels, reinforcing that FADE does not require privileged information to remain robust. Furthermore, this ablation clarifies the specific role of the $\mathcal{L}_{\text{Cost}}$ component: while $\mathcal{L}_{\text{Flood}}$ drives the primary exhaustion of the query budget, $\mathcal{L}_{\text{Cost}}$ is necessary for bypassing the temporal consistency filters (e.g., State Space Models) inherent in memory-heavy architectures like Samba. Without cost mimicry, the attack's impact on Samba is significantly diminished, whereas the inclusion of $\mathcal{L}_{\text{Cost}}$, even when guided by noisy predictions, successfully deceives the temporal updater into siphoning legitimate track identities.

Table 8. Ablation on Cost Mimicry in the TQF Loss (HOTA)

Attack	Dataset	MOTR	MOTRv2	MeMOTR	Samba	CO-MOT
TQF w/ GT	MOT17 | MOT20	45.90 | 42.63	39.29 | 29.64	41.41 | 57.67	45.53 | 46.85	37.26 | 33.37
TQF w/ Pred	MOT17 | MOT20	45.91 | 42.46	39.17 | 29.69	42.26 | 58.25	44.82 | 46.31	37.11 | 33.30
TQF w/o $\mathcal{L}_{\text{Cost}}$	MOT17 | MOT20	45.84 | 42.42	38.48 | 29.67	42.53 | 59.38	53.14 | 55.62	36.44 | 33.58

12.3. TQF Guided Physical Attacks

We further evaluate the efficacy of the FADE's TQF attack when optimizing the physical AAI and EAI perturbations. Tables 9 and 10 breakdown the performances on the MOT17 and MOT20 benchmarks, respectively.

Table 9. Simulated Physical FADE-TQF Attacks on MOT17.

MOT17 Dataset – Average scene density ∼21 objects/frame.								
Tracker	**Attack_Vector**	**HOTA↓**	**DetA↓**	**AssA↓**	**IDF1↓**	**IDR↓**	**IDP↓**	**IDSW↑**
MOTR	Clean	58.63	49.90	70.38	69.35	68.48	71.40	7.23
	TQF_AAI	**52.66**	**45.19**	**62.80**	**64.66**	**61.06**	**70.08**	**7.70**
	TQF_EAI	**42.74**	**48.15**	**39.04**	**47.68**	**42.49**	**54.77**	**8.87**
MOTRv2	Clean	59.96	49.15	74.71	71.99	60.89	89.68	1.75
	TQF_AAI	**52.50**	**43.08**	**65.32**	**65.06**	**52.48**	**87.28**	**2.64**
	TQF_EAI	**53.79**	**44.26**	**66.68**	**66.79**	**53.87**	**89.35**	**1.94**
MeMOTR	Clean	67.35	57.87	79.60	80.83	70.78	94.84	0.81
	TQF_AAI	**56.55**	**47.76**	**68.03**	**70.95**	**57.35**	**93.59**	**1.29**
	TQF_EAI	**59.80**	**51.07**	**71.09**	**74.76**	**62.02**	**94.74**	**1.01**
Samba	Clean	62.91	50.58	79.37	73.67	60.30	95.93	1.02
	TQF_AAI	**48.94**	**37.95**	**64.15**	**59.15**	**43.75**	**93.01**	**1.94**
	TQF_EAI	**56.67**	**45.88**	**71.01**	**69.16**	**54.59**	**95.37**	**1.14**
CO-MOT	Clean	58.16	46.22	74.87	69.87	57.21	91.97	1.83
	TQF_AAI	**50.80**	**40.02**	**65.98**	**63.15**	**49.45**	**89.66**	**2.96**
	TQF_EAI	**51.72**	**41.22**	**66.48**	**64.64**	**51.16**	**90.46**	**3.00**

12.3.1. Efficacy of TQF in Physical Constraints

The TQF attack, designed to exhaust the tracker's query budget with spurious, high-confidence temporal tracks, proves effective even under simulated physical attacks. On MOT17, the TQF-EAI attack consistently degrades HOTA

Table 10. Simulated Physical FADE-TQF Attacks on MOT20.

MOT20 Dataset – High Density: Avg. ∼150 objects/frame.								
Tracker	**Attack_Vector**	**HOTA↓**	**DetA↓**	**AssA↓**	**IDF1↓**	**IDR↓**	**IDP↓**	**IDSW↑**
MOTR	Clean	55.06	40.57	75.89	64.10	57.65	74.80	5.14
	TQF_AAI	**50.47**	**37.94**	**68.31**	**61.04**	**52.36**	**75.36**	**5.45**
	TQF_EAI	**41.97**	**36.05**	**52.00**	**48.81**	**38.48**	**68.60**	**6.60**
MOTRv2	Clean	59.56	44.74	80.71	69.55	54.90	96.81	0.73
	TQF_AAI	**53.75**	**40.34**	**72.93**	**64.76**	**49.18**	**97.02**	**0.80**
	TQF_EAI	**52.97**	**39.91**	**71.54**	**63.92**	**48.39**	**96.09**	**0.89**
MeMOTR	Clean	69.61	59.19	83.17	83.31	72.84	97.99	0.46
	TQF_AAI	**59.24**	**49.89**	**71.45**	**73.99**	**60.36**	**96.58**	**0.86**
	TQF_EAI	**62.29**	**52.95**	**74.41**	**77.68**	**64.72**	**97.75**	**0.57**
Samba	Clean	62.49	47.83	83.29	72.47	58.23	98.27	0.43
	TQF_AAI	**48.93**	**34.56**	**70.97**	**57.08**	**41.19**	**96.94**	**0.81**
	TQF_EAI	**55.29**	**41.71**	**74.79**	**66.22**	**50.62**	**98.27**	**0.51**
CO-MOT	Clean	64.31	52.30	80.35	78.01	65.43	97.78	0.45
	TQF_AAI	**57.40**	**46.52**	**72.00**	**72.23**	**57.77**	**97.75**	**0.60**
	TQF_EAI	**56.28**	**45.29**	**71.14**	**70.67**	**56.06**	**97.20**	**0.66**

scores by 10-16 points across all trackers. For the purely query-based MOTR, TQF-EAI reduces HOTA from 58.63 to 42.74, a signifcant 27% performance drop. A critical finding is the fragility of the Samba tracker to TQF-AAI. On MOT20, Samba's HOTA collapses from 62.49 to 48.93. The AAI-induced blur appears to interact destructively with the SSM's sequential scanning mechanism, allowing the flooding queries to easily overwhelm the state transitions, leading to massive detection loss (DetA drops from 47.83 to 34.56). In the high-density MOT20 dataset, the TQF attack is particularly powerful. Since the scene already contains ∼150 legitimate objects, the tracker's query budget is naturally near saturation. The TQF attack exploits this by pushing the system into failure with fewer adversarial perturbations than required in sparse scenes.

12.3.2. TMC vs. TQF in Simulated Physical Settings

Comparing the two attack modalities (Tables 3/4 vs. Tables 9/10) reveals distinct failure modes:

***(i)* Failure Mode Specificity (AssA vs. DetA):** The TMC attack consistently causes larger drops in AssA than DetA. For example, on MOTR (MOT17), TMC-EAI drops AssA by ∼31 points, while DetA remains relatively stable. This confirms TMC breaks the *temporal link*. Conversely, TQF attacks often degrade DetA and Recall (IDR) more severely. On MOT20, TQF-AAI drops Samba's DetA by ∼13 points. This confirms TQF acts as a *Denial-of-Service* attack, exhausting the query budget and forcing the tracker to discard and drop valid object detections entirely.

***(ii)* Robustness to Architecture:** MeMOTR, with its dedicated long-term memory bank, shows higher resilience to TQF than TMC. Its explicit memory module seemingly helps it distinguish between *flooding* noise and valid tracks better than it can handle the direct memory corruption of TMC. State-Space Models (Samba) appear uniquely vulnerable to TQF-AAI (Blur), suffering larger drops than purely Transformer-based models.

***(iii)* Attack Vector Synergy:** The sharp, high-frequency artifacts of the EAI (Glitch) attack appear suited for the precise feature-cutting required by TMC. The smeared, broad artifacts of the AAI (Blur) attack seem effective at generating the wide-area confusion necessary for TQF flooding, especially in trackers with localized attention windows.

In summary, while both attacks are effective, they compromise TBP trackers through orthogonal mechanisms: TMC acts as a *surgical strike* on identity integrity, while TQF acts as a *capacity overload* on the detection pipeline.

12.4. Long-Term Memory State Analysis

We analyze how the long-term tracking memory is compromised under FADE's TMC attack. The $\mathcal{L}_{\text{TMC}}$ adversarial loss is designed to fundamentally undermine the memory integrity in advanced TBP trackers, like MeMOTR [7], used for temporal robustness and long-term identity association. In Fig.9, by comparing the memory state before attack (PGD Step 0, clean) and at the final attack optimization step (PGD Step 149, fully attacked), we empirically validate the success of the attack in three critical dimensions.

12.4.1. Memory Distinctiveness (Self-Similarity Matrix)

This analysis (Fig.9.a plots) measures the self-similarity matrix ($\mathcal{M} \cdot \mathcal{M}^T$) of the stored track embeddings.

- **PGD Step 0 (Clean):** The strong, high-valued diagonal stripe (similarity ≈ 1.0) against orthogonal off-diagonal values confirms high identity integrity.
- **PGD Step 149 (Attacked State):** The diagonal feature is completely eliminated, vanishing into a uniform background of yellow-green noise. This demonstrates successful identity collapse, as all tracks are now nearly equally similar to one another.

12.4.2. Memory Embedding Statistics (Feature Stability)

This analysis (Fig.9.b plots) tracks the magnitude (L2 Norm) and stability (Std Dev) of the embeddings.

- **PGD Step 0 (Clean):** The L2 Norm (blue bars) is high and consistent, indicating robust, high-energy features across all stable tracks.
- **PGD Step 149 (Attacked State):** The L2 Norm remains relatively high but exhibits massive variability across the memory index. This visual chaos confirms the introduction of significant *Feature Distortion* and instability, which destroys the reliable structure needed for smooth temporal propagation.

12.4.3. Current-Memory Cross-Similarity

This analysis (Fig.9.c plots) is the ultimate measure of tracking failure, showing how current observations match against the historical memory bank.

- **PGD Step 0 (Baseline):** The presence of isolated, bright horizontal stripes indicates high confidence matching. Current frame queries find unambiguous, strong similarities against the relevant historical tracks.
- **PGD Step 149 (Attacked State):** The matrix is saturated with a diffuse, chaotic pattern. Distinct, high-

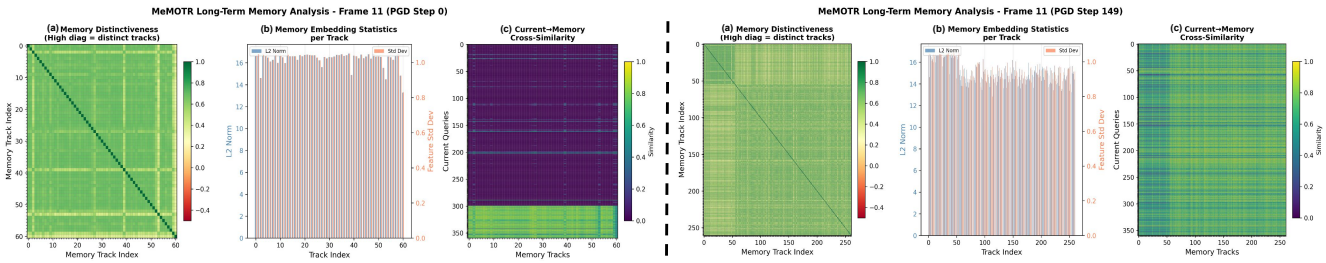

Figure 9. Analysis of Long-term Memory Integrity in TBP Trackers. The figure depicts the state of the MeMOTR [7]'s memory bank before (PGD Step 0) and after (PGD Step 149) the FADE attack, showing how FADE undermines the tracking identity memory mechanism.

confidence matches are eliminated. This demonstrates successful *Match Ambiguity*, as current queries now find low-to-medium similarity with a large number of corrupted memory tracks, forcing the tracker's assignment algorithm to fail (leading to ID switches and track terminations).

12.5. Analysis of Black-box Transferability

To evaluate the practical threat boundaries of FADE, we conduct a transferability analysis in a black-box setting. Following established protocols in the field [11, 40], we employ a three-frame attack window to standardize the adversarial budget. This setup allows us to move beyond the minimalist efficiency baseline of the single-frame white-box attacks presented in the main paper and investigate the *architectural persistence* of vulnerabilities across unseen targets.

Table 11. Black-box Transferability Check (HOTA) on MOT17.

Surrogate	Target → Clean HOTA	MOTR	MOTRv2	MeMOTR	Samba	CO-MOT
	None (Baseline)	58.63	59.96	67.35	62.91	58.16
MOTR	TMC | TQF	42.74 | 42.72	51.89 | 52.04	60.31 | 60.31	56.12 | 55.53	48.97 | 49.51
MOTRv2	TMC | TQF	42.76 | 42.79	37.08 | 27.98	60.65 | 60.11	55.94 | 56.56	48.09 | 44.78
MeMOTR	TMC | TQF	47.35 | 46.26	54.11 | 53.76	46.01 | 29.10	52.10 | 52.43	52.16 | 52.44
Samba	TMC | TQF	47.60 | 47.72	54.24 | 53.88	55.20 | 56.45	43.22 | 36.30	52.68 | 52.13
CO-MOT	TMC | TQF	42.76 | 42.74	51.84 | 49.62	60.53 | 60.32	55.90 | 56.29	32.48 | 27.03

As shown in Table 11, we observe significant transferability when the surrogate and target trackers share a common *architectural lineage*. Our results reveal three primary transferability clusters:

- *Query-Update Lineage:* Strong transferability is observed between MOTR and MOTRv2, which share similar recurrent query-update logic.
- *Detector Priors:* MOTRv2 and CO-MOT exhibit high mutual transferability due to their reliance on similar external detection priors.
- *Memory Modules:* For specialized memory-augmented architectures like MeMOTR and Samba, using a surrogate with a comparable temporal memory module maximizes the HOTA degradation.

Notably, MOTRv2 emerges as the most effective general-purpose surrogate for black-box deployment, while memory-specific surrogates are required to effectively disrupt the most complex TBP architectures. These findings proves that FADE exploits fundamental structural dependencies inherent to the TBP paradigm rather than model-specific overfitting.

12.6. FADE under Common Defenses

To evaluate the robustness of the FADE attack strategies against standard input transformation defenses, we employ three input transformation based defense techniques: Color Jittering (CJ), Spatial Smoothing (SS), and Gaussian Noise (GN). These methods aim to disrupt the specific adversarial perturbations generated by the attack before the input reaches the MOT tracker.

Defense 1: Color Jittering (CJ) Adversarial perturbations often rely on precise pixel values to mislead the model's gradient. Color Jittering disrupts these dependencies by randomly altering the photometric properties of the input frames. We apply random transformations to the brightness, contrast, saturation, and hue of the adversarial images x_{adv}. The transformation can be formulated as a stochastic function $\mathcal{T}_{CJ}(\cdot)$:

$$x_{def} = \mathcal{T}_{CJ}(x_{adv}; \lambda_b, \lambda_c, \lambda_s, \lambda_h)$$

where λ represents the jitter factors. In our experiments, we randomly sample brightness, contrast, and saturation factors from $[1-\delta, 1+\delta]$ and hue from $[-\delta, \delta]$, setting $\delta = 0.2$. This forces the tracker to rely on structural features rather than specific pixel intensities that may have been compromised by the attack.

Defense 2: Spatial Smoothing (SS) Since adversarial noise typically manifests as high-frequency fluctuations imperceptible to the human eye, Spatial Smoothing acts as a low-pass filter to mitigate these artifacts. We utilize a Gaussian blur kernel to convolve the input image, effectively smoothing out the sharp gradients introduced by the attack optimization process. The smoothed image x_{SS} is obtained via:

$$x_{SS} = x_{adv} * G_{k,\sigma}$$

where G is a Gaussian kernel of size $k \times k$ with standard deviation σ. We evaluate robustness using a kernel size of $k = 3$ and $\sigma = 0.5$. This defense tests the attack's ability to survive the removal of its high-frequency components.

Table 12. FADE-TMC Performances under Defenses on MOT17.

MOT17 Dataset – Average scene density ∼21 objects/frame.									
Tracker	**Attack_Vector**	**Defense**	**HOTA**	**DetA**	**AssA**	**IDF1**	**IDR**	**IDP**	**IDSW**
MOTR	Clean	None	58.63	49.90	70.38	69.35	68.48	71.40	7.23
	$FADE_{TMC}$	**CJ**	**46.00**	**47.45**	**45.88**	**51.67**	**46.68**	**58.57**	**9.67**
	$FADE_{TMC}$	**GN**	**46.38**	**47.90**	**46.22**	**51.87**	**46.63**	**59.05**	**9.51**
	$FADE_{TMC}$	**SS**	**46.02**	**47.88**	**45.54**	**51.48**	**46.13**	**58.82**	**9.56**
	$FADE_{TMC}$	**None**	**45.89**	**51.36**	**42.18**	**50.45**	**46.35**	**55.79**	**8.77**
MOTRv2	Clean	None	59.96	49.15	74.71	71.99	60.89	89.68	1.75
	$FADE_{TMC}$	**CJ**	**53.35**	**42.30**	**69.14**	**63.96**	**52.95**	**83.62**	**3.25**
	$FADE_{TMC}$	**GN**	**52.61**	**41.30**	**68.87**	**63.15**	**51.72**	**83.65**	**3.56**
	$FADE_{TMC}$	**SS**	**52.95**	**41.95**	**68.82**	**63.58**	**52.38**	**83.97**	**3.31**
	$FADE_{TMC}$	**None**	**46.76**	**37.63**	**59.44**	**56.22**	**42.80**	**83.89**	**3.83**
MeMOTR	Clean	None	67.35	57.87	79.60	80.83	70.78	94.84	0.81
	$FADE_{TMC}$	**CJ**	**57.13**	**53.79**	**61.78**	**66.96**	**56.44**	**82.98**	**2.80**
	$FADE_{TMC}$	**GN**	**57.03**	**53.26**	**62.25**	**66.78**	**56.17**	**83.11**	**2.86**
	$FADE_{TMC}$	**SS**	**57.12**	**53.61**	**62.01**	**66.67**	**56.17**	**82.60**	**2.81**
	$FADE_{TMC}$	**None**	**41.56**	**35.74**	**49.18**	**51.60**	**37.61**	**84.07**	**4.63**
Samba	Clean	None	62.91	50.58	79.37	73.67	60.30	95.93	1.02
	$FADE_{TMC}$	**CJ**	**50.55**	**46.44**	**56.02**	**58.11**	**46.06**	**80.74**	**3.19**
	$FADE_{TMC}$	**GN**	**50.97**	**46.95**	**56.51**	**58.31**	**46.44**	**80.72**	**3.24**
	$FADE_{TMC}$	**SS**	**50.43**	**46.61**	**55.66**	**57.70**	**45.89**	**79.90**	**3.31**
	$FADE_{TMC}$	**None**	**48.04**	**45.19**	**51.93**	**56.01**	**44.01**	**78.74**	**3.63**
CO-MOT	Clean	None	58.16	46.22	74.87	69.87	57.21	91.97	1.83
	$FADE_{TMC}$	**CJ**	**52.61**	**40.54**	**70.57**	**62.51**	**49.77**	**88.24**	**2.47**
	$FADE_{TMC}$	**GN**	**53.54**	**41.69**	**71.23**	**63.81**	**51.07**	**89.64**	**1.91**
	$FADE_{TMC}$	**SS**	**52.86**	**40.75**	**70.70**	**63.09**	**50.82**	**87.54**	**2.68**
	$FADE_{TMC}$	**None**	**41.73**	**31.82**	**55.89**	**50.34**	**39.23**	**72.34**	**10.94**

Table 13. FADE-TQF Performances under Defenses on MOT17.

MOT17 Dataset – Average scene density ∼21 objects/frame.									
Tracker	**Attack_Vector**	**Defense**	**HOTA**	**DetA**	**AssA**	**IDF1**	**IDR**	**IDP**	**IDSW**
MOTR	Clean	None	58.63	49.90	70.38	69.35	68.48	71.40	7.23
	$FADE_{TQF}$	**CJ**	**47.00**	**47.47**	**47.87**	**53.09**	**47.88**	**60.24**	**9.17**
	$FADE_{TQF}$	**GN**	**47.03**	**47.92**	**47.51**	**53.19**	**48.23**	**60.04**	**9.69**
	$FADE_{TQF}$	**SS**	**47.18**	**47.26**	**48.43**	**53.21**	**48.15**	**60.08**	**9.47**
	$FADE_{TQF}$	**None**	**45.90**	**51.40**	**42.17**	**50.51**	**46.40**	**55.86**	**8.73**
MOTRv2	Clean	None	59.96	49.15	74.71	71.99	60.89	89.68	1.75
	$FADE_{TQF}$	**CJ**	**53.19**	**42.42**	**68.55**	**63.96**	**52.70**	**84.00**	**3.02**
	$FADE_{TQF}$	**GN**	**53.18**	**42.09**	**69.06**	**63.68**	**52.51**	**84.13**	**3.31**
	$FADE_{TQF}$	**SS**	**52.77**	**42.14**	**67.93**	**63.44**	**52.37**	**83.19**	**3.89**
	$FADE_{TQF}$	**None**	**39.29**	**31.68**	**49.96**	**49.02**	**35.87**	**78.77**	**5.65**
MeMOTR	Clean	None	67.35	57.87	79.60	80.83	70.78	94.84	0.81
	$FADE_{TQF}$	**CJ**	**62.97**	**54.47**	**73.96**	**75.44**	**63.93**	**92.69**	**1.43**
	$FADE_{TQF}$	**GN**	**63.39**	**54.74**	**74.53**	**76.10**	**64.67**	**93.08**	**1.35**
	$FADE_{TQF}$	**SS**	**62.91**	**54.43**	**73.88**	**75.25**	**63.72**	**92.61**	**1.43**
	$FADE_{TQF}$	**None**	**41.41**	**34.83**	**50.03**	**51.41**	**37.07**	**85.38**	**4.31**
Samba	Clean	None	62.91	50.58	79.37	73.67	60.30	95.93	1.02
	$FADE_{TQF}$	**CJ**	**55.31**	**45.90**	**68.17**	**64.25**	**50.79**	**90.41**	**1.93**
	$FADE_{TQF}$	**GN**	**56.16**	**46.60**	**69.16**	**65.46**	**52.14**	**90.87**	**1.78**
	$FADE_{TQF}$	**SS**	**54.95**	**45.78**	**67.60**	**63.99**	**50.71**	**90.09**	**1.93**
	$FADE_{TQF}$	**None**	**45.53**	**32.71**	**64.37**	**52.71**	**37.62**	**91.33**	**2.24**
CO-MOT	Clean	None	58.16	46.22	74.87	69.87	57.21	91.97	1.83
	$FADE_{TQF}$	**CJ**	**52.65**	**40.54**	**70.50**	**62.88**	**50.72**	**87.19**	**2.99**
	$FADE_{TQF}$	**GN**	**53.42**	**41.29**	**71.05**	**63.81**	**50.82**	**89.67**	**1.91**
	$FADE_{TQF}$	**SS**	**52.44**	**40.13**	**70.71**	**62.15**	**49.84**	**87.51**	**2.50**
	$FADE_{TQF}$	**None**	**37.26**	**27.43**	**51.93**	**44.84**	**32.88**	**74.66**	**9.50**

Defense 3: Gaussian Noise (GN) The Gaussian Noise defense introduces random stochasticity to the input, aiming to disrupt the precise alignment of the adversarial perturbation. While adding noise seems counter-intuitive, it is a foundational concept in *Randomized Smoothing*, where the classifier is forced to be robust within a local neighborhood of the input. We inject additive white Gaussian noise η into the adversarial input:

$$x_{GN} = x_{adv} + \eta, \quad \text{where } \eta \sim \mathcal{N}(0, \sigma_{gn}^2)$$

We set the noise standard deviation $\sigma_{gn} = 0.1$. This defense effectively drowns out small-magnitude adversarial perturbations, testing whether the attack features are distinct enough to persist through random interference.

12.6.1. Analysis of Robustness and Defense Efficacy

To assess the robustness of the FADE framework, we evaluate three standard input transformation defenses: Color Jitter (CJ), Gaussian Noise (GN), and Spatial Smoothing (SS). Tables 12 and 13 detail the performance of five TBP-based trackers under these defenses against FADE-TMC and FADE-TQF, respectively. We consider the same attack implementation details for one attacked frame only.

***(i)* The Recovery Gap.** A consistent pattern across all experiments is the existence of a *Recovery Gap*: The performance difference between the Defended state and the Clean baseline. On the MOTR tracker, defenses provide negligible benefit against the TMC attack. As shown in Table 12, the undefended FADE-TMC attack reduces HOTA to 45.89. Applying Gaussian Noise (GN) only raises this to 46.38, and Spatial Smoothing (SS) to 46.02. This indicates that for pure query-propagation architectures, the adversarial perturbation is deeply embedded in the feature representation and cannot be removed by simple pre-processing. In contrast, the MeMOTR tracker demonstrates some recoverability, particularly against TQF. In Table 13, the TQF attack drops MeMOTR's HOTA to 41.41. However, defenses are able to restore this to ∼63.0 (e.g., GN yields 63.39), bringing it quite closer to the clean baseline of 67.35. This suggests that MeMOTR's long-term memory bank provides a *stabilizing anchor* that, when aided by input sanitization, can filter out transient adversarial noise.

***(ii)* Defense Invariance.** An analysis of the specific defense modalities reveals a high degree of performance invariance. Across almost all trackers and attack vectors, the variation in HOTA scores between CJ, GN, and SS is statistically insignificant (typically < 0.5 points). For example, on MOTRv2 under FADE-TQF (Table 13), the HOTA scores for CJ, GN, and SS are 53.19, 53.18, and 52.77, respectively. This invariance implies that the FADE attack does not rely on a single fragile feature modality (such as high-frequency noise alone or specific color triggers). Instead, the adversarial optimization converges to a robust perturbation that permeates multiple feature domains, making it resilient to any single type of input transformation.

***(iii)* Vulnerability of State-Space Models.** The results highlight a fragility in the Samba tracker regarding defense efficacy. While MeMOTR recovers ∼83% of its lost performance under defenses against TQF, Samba shows significantly lower elasticity. Under FADE-TMC (Table 12), Samba drops to 48.04 HOTA. The best-performing defense (GN) only restores this to 50.97, leaving a massive gap from the clean baseline of 62.91. This suggests that for State-Space Models, the adversarial corruption interferes with the sequential state dynamics in a way that frame-level preprocessing cannot correct. Once the state transition is effectively jammed by the attack, cleaning the input pixels is

insufficient to reset the tracking logic.

***(iv)* Divergent Defense Efficacy: TMC vs. TQF.** A cross-comparison of Tables 12 and 13 reveals a fundamental divergence in how the two attacks respond to input sanitization. The TQF attack operates by generating fake queries, spurious high-confidence detections, designed to exhaust the tracker's budget. The results show that these fake queries are relatively fragile. On the MeMOTR tracker, defenses recover the HOTA score from 41.41 (undefended) to ∼63.0. This represents a recovery of nearly 83% of the performance drop. Spurious objects are often triggered by high-frequency adversarial noise patterns that mimic object textures. Standard defenses like Spatial Smoothing and Gaussian Noise can disrupt these high-frequency triggers, causing the detector to suppress the fake queries. Once the fakes are removed, the query budget is relieved, and the tracker recovers. In contrast, the TMC attack proves exceptionally resistant to defenses. On the same MeMOTR tracker, defenses against TMC only recover the HOTA score from 41.56 to ∼57.03. Recovery saturates at just 57%, leaving a permanent performance gap. On MOTR, the recovery is negligible ($< 1\%$). TMC does not rely on creating new, fragile objects. Instead, it subtly shifts the embeddings of *existing, valid* objects within the feature manifold to break their temporal links. This perturbation is semantically aligned with the track identity features (often lower-frequency patterns) rather than being mere surface noise. Consequently, simple input filters (CJ, GN, SS) cannot remove the adversarial signal without also degrading the legitimate object features, rendering the defense ineffective. This divergence confirms that while TQF acts as a *surface-level* capacity attack (mitigatable via pre-processing), TMC acts as a *deep-feature* identity attack that requires architectural hardening to defeat.

12.7. Practicality, Realism, and Future Work

The practical viability of FADE is supported by its computational efficiency and its grounding in physically calibrated sensor models. On modern hardware, the PGD optimization process requires approximately 3 s on an NVIDIA RTX 3090 and reaches $\sim$ 1 s on an A100 per frame. While per-frame optimization is useful for characterizing vulnerability bounds, the high transferability observed in Table 11 suggests that pre-computed perturbations generated on a surrogate model can enable *zero-latency online deployment*. By leveraging these pre-computed adversarial perturbations, an adversary can bypass the need for real-time optimization entirely, facilitating immediate state-poisoning of the target tracker. Regarding realism, our experiments utilize sensor-level simulations (AAI and EAI) that are directly calibrated to real-world camera profiles [10, 12, 14, 41], as detailed in Section 10 of the Appendix. While we acknowledge that closed-loop hardware-in-the-loop validation remains a distinct engineering challenge involving variable environmental factors, this study establishes the foundational algorithmic feasibility and the prerequisite failure physics necessary for such future benchmarks. To further enhance the robustness of these threats, future work may explore *ensemble learning techniques* [30, 35] to generate universal adversarial perturbations. By optimizing across a diverse pool of TBP architectures simultaneously, we posit that it may be possible to overcome current architectural lineage constraints and develop truly cross-paradigm adversarial vectors that generalize to entirely unknown tracking systems.