WHEN ADAPTATION BECOMES VULNERABILITY: DYNAMIC PATCHES OPEN THE TEMPORAL ATTACK SURFACE OF ONLINE TRACKERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Visual object tracking has witnessed a paradigm shift from offline static matching to online adaptive temporal learning, where trackers achieve robustness by continuously learning from observations and dynamically updating their target models. However, we reveal a critical flaw hidden beneath this learning-from-observation flexibility: a static adversarial patch is progressively memorized and assimilated by the tracker's online update mechanism, so the attack signal is not defeated but absorbed, and its efficacy decays sharply as memory accumulates. Ours **DyPatch** is the first to characterize this categorical mismatch between static carriers and learning systems as a directly exploitable attack surface: the tracker's temporal learning process itself, rather than its perceptual output. Unlike traditional attacks that target the perceptual output, this attack surface enables simultaneous manipulation of both the model's semantic representation and its online evolution trajectory. Accordingly, we propose a dynamic adversarial patch framework, which, through temporal switching and adversarial prior inheritance, produces dynamic patches that precisely hijack the tracker, trapping its online update into a vicious cycle of the more it updates, the more it errs. Experiments across six trackers and four benchmarks demonstrate that dynamic patches sustain effective attacks where static patches invariably fail, and successfully transfer to the physical world.

1 INTRODUCTION

Online adaptive trackers (Cai et al., 2024; Guo et al., 2025; Ding et al., 2025b) maintain and continuously update an internal target model across frames, adapting to appearance variation, occlusion, and long-term drift. This online adaptation is the source of their state-of-the-art robustness, and it has made them the backbone of safety-critical systems such as autonomous driving and intelligent surveillance (Cai et al., 2025; Ding et al., 2025a; Li et al., 2026). In this work, we argue that this same mechanism, designed as the online tracker's greatest strength, is also its most fundamental vulnerability, exposing a new attack surface that adversarial attacks (Yan et al., 2020; Zhao et al., 2023; Xiang et al., 2026a;b) on trackers have so far overlooked.

Back to previous pioneers' adversarial attacks on visual object tracking. They can be broadly categorized into perturbation-based and patch-based approaches. The former (Jia et al., 2021; Fu et al., 2022; Jia et al., 2025; Gao et al., 2025; Zhao et al., 2026) corrupt the model by imposing global or local pixel-level perturbations on the image; despite their strong misleading capability, they are difficult to deploy in the real world. The latter (Ding et al., 2021; Xuesong et al., 2021; Xiang et al., 2025a;b) instead interfere by pasting localized adversarial patterns onto the target surface. Owing to their printability, easy attachment, and reusability, patch-based attacks have become the mainstream form of physical-domain attacks, offering high feasibility in real-world scenarios and thus being better suited for real-world security verification and deployment.

Although patch-based attacks have advanced the attack perspective from the digital to the physical domain (Ding et al., 2021; Xie et al., 2026), existing methods universally adopt static patches as the attack carrier, which fail to reach the attack surface unique to tracking as a sequential task and are therefore ill-suited for online trackers. Specifically, static patches are incapable of dynamic evolution. Online trackers continuously learn and memorize such invariant historical cues, so the

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

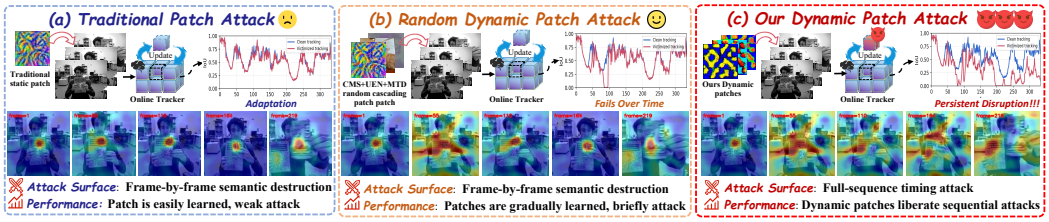


Figure 1: For online trackers, both static patch attacks and randomly combined dynamic patch attacks have gradually been absorbed. Our dynamic patches present great attack performance, which has revealed a brand-new attack surface for online trackers: full-sequence timing attacks.

adversarial signals are readily assimilated into the target's appearance representations. Consequently, attack performance degrades drastically as the tracker accumulates temporal memory. As shown in Fig. 1(a), after a brief period of deceptive interference, static patches are rapidly incorporated by the tracker's online update mechanisms (*e.g.*, dynamic template replacement) and even converted into prior knowledge that facilitates stable tracking. To tackle the aforementioned limitations, an intuitive and feasible remedy is to cascade multiple highly adversarial static patches for cyclic deployment.

Although such a simple cascading strategy can generate plausible adversarial carriers (*i.e.*, dynamic patches), its underlying attack logic is inherently constrained by a static paradigm and fails to fully exploit the potential of temporal attacks (as illustrated in Fig. 1(b)). While this strategy suppresses the memory evolution of trackers to a certain extent, it still struggles to deliver effective adversarial attacks. Two core reasons account for this limitation: **First, ignore the attacks of dynamic patches in the temporal dimension.** Conventional patch combination schemes only prioritize the attack intensity within individual frames, without leveraging the discontinuities introduced by patch switching to construct abrupt temporal attacks. Rooted in this static mindset, existing attacks are confined solely to the spatial semantic domain and disregard trackers' reliance on temporal smoothness via online update mechanisms. Consequently, trackers can mitigate adversarial impacts through moving average operations. **Second, lack of a continuous adversarial evolution mechanism among patches.** Without efficient transfer of adversarial information and coordinated collaboration across multiple patches, attack gradients cannot accumulate temporally to build an amplified adversarial tendency that exceeds the temporal memory threshold of trackers. This prevents sustained, stable adversarial suppression. In summary, neither static patch nor dynamically combined random patches can inflict persistent, accumulative adversarial interference on the memory module. Online trackers will eventually memorize such fixed attack patterns, leading to degraded overall attack performance.

To tackle the above limitations, we discard the obsolete framework that relies on static patches to deliver temporal attacks, and innovatively propose a genuine temporal dynamic attack paradigm. Specifically, we adopt cyclically repeated dynamic patches as adversarial carriers paired with tailored dynamic attack logic to corrupt the trackers' historical memory and target semantic perception, thus achieving potent adversarial effects against online trackers, as visualized in Fig. 1(c). The detailed dynamic attack logic consists of three core designs, as shown in Fig. 2. **First, we introduce the temporal switching attack optimization.** Concretely, we explicitly boost the adversarial strength of patches at the tail and head frames of each cycle. Such abrupt transitions disrupt the trackers' short-term memory and raise the difficulty for trackers to summarize consistent target patterns from historical observations. **Second, an adversarial prior inheritance framework is devised.** Equipped with learnable parameters, each current patch inherits valid adversarial priors accumulated from previous attacks while adaptively injecting incremental perturbations customized to the real-time tracker status. Under this mechanism, the patch sequence is no longer composed of isolated noise signals; instead, it generates continuously aggregated misleading bias. It gradually drives the tracker's historical memory toward a distribution far deviated from the genuine target, trapping the online update pipeline into a vicious cycle where model predictions degrade with every iterative update. Coupled with a difficulty-aware adversarial loss that dynamically adjusts optimization weights according to the attack difficulty of individual samples, our method keeps the tracker's memory system under persistent high-load instability. **Third, we propose a spatio-temporal degradation augmentation strategy.** To strengthen the robustness of dynamic patches in complex physical scenarios, we jointly model and adversarially optimize diverse real-world temporal degradations, including frame rate jitter, motion blur, illumination variation, viewpoint shift. As a result, the crafted

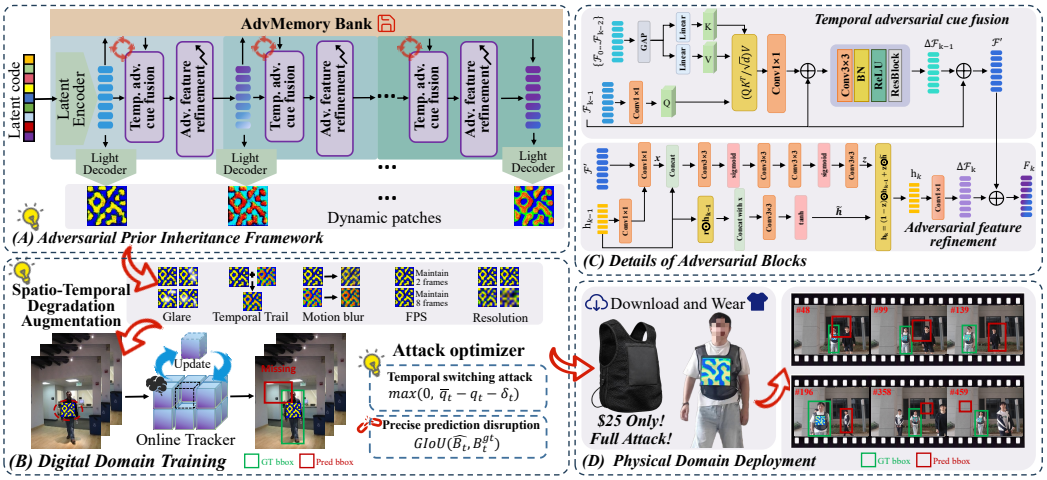


Figure 2: The overall framework of Our DyPatch. With the temporal switching attack loss, dynamic patches are formed through adversarial prior inheritance framework, which is combined with spatio-temporal degradation augmentation to enhance.

attack patterns maintain stable and strong adaptability and evasion capability in physical real-world environments. In summary, we make the following contributions:

- We expose a previously overlooked attack surface rooted in the adaptive mechanism rather than the perceptual output, and propose the dynamic adversarial patches paradigm **(DyPatch)**, which disrupts online trackers by continuously evolving the dynamic patches to hijack the target trackers.
- Adopting periodic dynamic patches, the temporal switching attack breaks the tracker's short-term memory equilibrium. The adversarial prior inheritance framework with temporal adversarial cue fusion module and adversarial feature refinement module further accumulates misleading biases and diverts the overall distribution away from the genuine target trajectory.
- We design a spatio-temporal degradation augmentation strategy, so that dynamic patches can be better extended from the digital to the physical world.
- Experimental results show that our dynamic patches can efficiently fool online trackers even normal trackers in standard datasets and real scenes.

2 Related Work

2.1 Online Visual Object Trackers

Visual object tracking aims to localize a target in consecutive frames given an initial bounding box. Existing trackers can be broadly categorized into offline and online ones. Offline trackers rely on a static model without learning from subsequent frames. For example, SiamFC++ (Xu et al., 2020) adopts a Siamese architecture for efficient template matching, while AsymTrack (Zhu et al., 2025) improves template discriminability through asymmetric feature fusion. Despite their relatively efficiency, these methods cannot adapt to substantial appearance changes and long-term variations, as they ignore historical observations. Online trackers instead continuously update target representations from each new observation, enabling more effective temporal adaptation. Classic method DiMP (Bhat et al., 2019) performs discriminative online learning with historical samples to continuously optimize the target model. STARK (Yan et al., 2021) introduces spatio-temporal attention to capture dependencies across frames, while ODTrack (Zheng et al., 2024) formulates tracking as temporal token propagation to preserve historical context. More recently, SpikeTrack (Zhang et al., 2026) further incorporates memory mechanisms into spike-driven tracking for efficient long-term adaptation. Although online adaptation improves robustness, it also creates a security concern: since the target representation is repeatedly updated, adversarially manipulated frames may corrupt not only the current prediction but also subsequent updates, accumulating

persistent errors. However, the security implications of these temporal adaptation mechanisms have received limited attention.

2.2 Adversarial Attacks on Visual Object Trackers

Adversarial attacks on trackers (Lou et al., 2023; Zhou et al., 2023; Huang et al., 2026) can be broadly categorized into perturbation-based and patch-based attacks. The former directly optimize subtle perturbations in the image space. Earlier method IoU attack (Jia et al., 2021) iteratively optimizes the IoU loss to craft temporally coherent perturbations, while PANC (Gao et al., 2025) recently compresses perturbation patch-wise to improve stealthiness and attack efficiency. Despite their strong misleading capability, the above methods are sensitive to imaging conditions and cannot be physically deployed. Instead, patch-based attacks provide a more practical alternative by modifying the target's visible appearance with localized adversarial patterns. MTD (Ding et al., 2021) introduces texture-based physical attacks to disrupt target feature matching, and UEN (Xuesong et al., 2021) provides a unified framework for digital and physical attacks. BAPGNet (Jia et al., 2023) generates adversarial patches to increase feature discrepancies between template and search regions. CMS (Xiang et al., 2025a), originally developed for cross-modal tracking, further improves patch adaptability across environments and modalities. Nevertheless, existing patch-based attacks remain confined to static carriers and cannot effectively attack online trackers. This motivates us to explore a dynamic paradigm better aligned with the temporal nature of tracking, thereby uncovering a new attack surface.

3 Method

3.1 Problem Formulation

Conventional visual object trackers predict bounding boxes $\hat{\mathcal{B}}_t = \mathcal{T}(z, x_t|\theta)$ solely from the template z and search frame $x_t \in \mathcal{X}$ with fixed parameters θ. In contrast, the online tracker maintains an internal target state S_t and updates it during tracking through a learning operator Φ, *i.e.*, $S_{t+1} = \Phi(S_t, \phi_i)$, where ϕ_i denotes recent the observations used for the update, and the prediction $\hat{\mathcal{B}}_t = \mathcal{T}(z, x_t|S_t, \theta)$ is read out from the state S_t. This additional state S_t reveals that the attack surface of an online tracker extends beyond the perceptual matching between z and x_t.

A static patch p_s, however, fails to exploit this channel. When p_s is overlaid on the observations, the update becomes $S_{t+1} = \Phi(S_t, \{\phi_i + \phi(p_s)\}) \to S^*$. Since Φ repeatedly assimilates the same stationary signal, S_t gradually converges to a fixed state S^* that has absorbed p_s, so the attack is neutralized rather than defeated. In the extreme case, p_s is learned as a new clue ϕ_{i+1} that facilitates stable tracking. We therefore formulate the attack as optimizing a dynamic patch set $\mathcal{P} = \{p_d\}_{d=1}^{D}$ that continuously poisons the state update in the latent space, rather than a single static patch. An attack is effective if it maximizes the cumulative tracking error while driving the internal state S_t away from any fixed point. Considering that GIoU can accurately express the overlap between the $\hat{\mathcal{B}}_t$ and the $\mathcal{B}_t^{gt}$, we also design adversarial prior inheritance framework $\mathcal{G}(\cdot;\theta_g)$ to generate $\{p_d\}_{d=1}^{D}$. The above optimization can be formulated as:

$$\arg\min_{\theta_g} \sum_{t=1}^{T} GIoU\Big(\mathcal{T}\big(z, x_t^{adv} \mid S_t, \theta\big),\ \mathcal{B}_t^{gt}\Big),\quad s.t.\ \ x_t^{adv} = x_t \odot (1 - M_p) + p_{k(t)} \odot M_p, \quad (1)$$

where $k(t) = t \bmod D$ selects the patch applied at frame t, and M_p is the patch mask. The second constraint reveals that the attack operates in the latent state space: at each frame, the patch feature $\phi(p_{k(t)})$ is injected into the state update Φ. Since this injected feature varies across frames, S_t never converges to a fixed point and is instead driven persistently away from the true target distribution. The attack thus propagates through S_t to corrupt every subsequent prediction. To fully exploit the temporal switching information and enable adversarial information to be inherited and evolved across patches, we first design a Temporal Switching Attack loss $\mathcal{L}_{tsa}$ to guide the optimization of the patches. We then introduce an Adversarial Prior Inheritance architecture $\mathcal{G}(\cdot;\theta_g)$ to form the dynamic patches $\{p_d\}_{d=1}^{D}$. Finally, we design a Spatio-Temporal Degradation Augmentation strategy ξ to guarantee the robustness of the patches in the physical world.

3.2 TEMPORAL SWITCHING ATTACK

In contrast to static patches, dynamic patches introduce the temporal dimension as an additional attack surface. Specifically, we exploit the discontinuity induced by patch switching to progressively accumulate or disrupt the tracker's persistent understanding of the target. Online trackers rely on temporal smoothness to maintain a consistent target representation, so a static patch is gradually absorbed through moving-average operations. Besides, because S_t is recursively updated, a transition $p_i \rightarrow p_{i+1}$ can alter both the immediate prediction and the subsequent tracking trajectory. The abrupt feature jumps between successive patches, break this smoothness: the tracker can no longer distill a stable representation from its historical observations, and the adversarial effect accumulates over time instead of being averaged away. More precisely, we employ GIoU to measure the attack strength, which we term the attack quality $q(\cdot)$. Since each patch is maintained for L frames, the frames surrounding the transition from p_i to p_{i+1} form a switching window. To fully exploit the attack gain along the temporal dimension, we enforce the attack quality after switching to exceed the average attack quality $\bar{q}_t = \frac{1}{L}\sum_{l=t-L}^{t-1} q_l$ within the switching window. Furthermore, since the cyclic patch pattern keeps the tracker from memorizing any fixed carrier, we additionally boost the attack quality when the cycle loops back from the last patch to the first. The overall formulation is shown as follows:

$$\mathcal{L}_{total} = \mathcal{L}_{atk} + \mathcal{L}_{tsa} = \sum_{t\in\mathcal{X}} GIoU\big(\hat{\mathcal{B}}_t, \mathcal{B}_t^{gt}\big) + \lambda_{switch} \sum_{t\in\mathcal{X}} \max\big(0, \bar{q}_t - q_t - \delta_t\big). \tag{2}$$

Notably, we set a small threshold δ_t during the attack. Since the cyclically repeated patches risk being memorized by the tracker, we raise the switching threshold when the last patch transitions back to the first patch. Specifically, ordinary transitions use $\delta_t = 0.03$, whereas the cyclic transition $p_D \rightarrow p_1$ uses $\delta_t = 0.06$ and a larger λ_{switch}. We set $L = 2$ and jointly optimize both losses. $\mathcal{L}_{tsa}$ turns patch changes into sustained state disturbances rather than isolated frame-level perturbations. More explanation on temporal switching attack are shown in Appendix A .

3.3 ADVERSARIAL PRIOR INHERITANCE

Constrained schemes combined with vanilla generators fail to yield robust dynamic patches. Motivated by this limitation, we present the Adversarial Prior Inheritance framework, which comprises two core modules: temporal adversarial cue fusion and adversarial feature refinement.

Temporal Adversarial Cue Fusion To form adversarial patches with progressively increasing attack strength by $\mathcal{L}_{tsa}$, we design a temporal adversarial cue fusion module $\mathcal{C}u(\cdot)$ that enriches the effective adversarial cues $\mathcal{F}$ accumulated from preceding patches. By aggregating these cues, each subsequent patch inherits and amplifies the attack capability of its predecessors, so the patch sequence escalates the adversarial intensity gradually, keeping the tracker under ever-increasing pressure while preventing it from adapting to any single pattern. Specifically, the preceding cues $\mathcal{F}_k$ provides the query, while globally pooled historical features banks $\mathcal{M}_b = \{\mathcal{F}_1, \dots \mathcal{F}_{k-1}\}$ provide the keys and values:

$$H_{k-1} = Softmax([GAP(F_0), \dots, GAP(F_{k-2})]), \tag{3}$$

$$\mathbf{Q} = g_q(F_{k-1}), \qquad \mathbf{K} = g_k(H_{k-1}), \qquad \mathbf{V} = g_v(H_{k-1}). \tag{4}$$

The retrieved prior and a lightweight adapter yield the first residual update:

$$\begin{aligned} \mathcal{F}_{\text{attn}} &= \mathcal{F}_{k-1} + g_o\left[Softmax\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}\right], \\ \Delta\mathcal{F}_{k-1} &= Adapter(\mathcal{F}_{\text{attn}}), \qquad \mathcal{F}' = \mathcal{F}_{k-1} + \Delta\mathcal{F}_{k-1}. \end{aligned} \tag{5}$$

When $\mathcal{M}_b$ is empty, we set $\mathcal{F}_{\text{attn}} = \mathcal{F}_{k-1}$. The adapter consists of a 3×3 convolution, batch normalization, ReLU, and a residual block. It converts the retrieved historical adversarial cues into a patch-specific residual without replacing the preceding adversarial feature.

Adversarial Feature Refinement To undermine the continual learning mechanism of online trackers, we further refine the adversarial prior inherited from the previous patch and synthesize a new patch by adversarial feature refinement framework $\mathcal{R}e(\cdot)$. The inherited adversarial prior provides information that enables early-stage attack effectiveness, yet further information aggregation is still required to achieve incremental perturbations. As such, the patch sequence is no longer a set of isolated noises

but an evolving carrier that keeps the tracker's target representation persistently unstable and prevents it from memorizing any fixed patterns. Specifically, we concatenate adversarial cue $\mathcal{F}'$ that worth mention with the projected hidden state h_{k-1}:

$$X_k = Proj(\text{concat}(\mathcal{F}', W_h(h_{k-1}))), \tag{6}$$

$$z_k = \sigma(W_z * \text{concat}(X_k, h_{k-1})), \qquad r_k = \sigma(W_r * \text{concat}(X_k, h_{k-1})), \tag{7}$$

$$\widetilde{h}_k = \tanh(W_c * \text{concat}(X_k, r_k \odot h_{k-1})), \qquad h_k = (1 - z_k) \odot h_{k-1} + z_k \odot \widetilde{h}_k, \tag{8}$$

where $\text{concat}(\cdot)$ and $*$ denote channel-wise concatenation and convolution, respectively. The recurrent state produces a second residual, after which a shared decoder maps all features, to patches:

$$\Delta\mathcal{F}_k = W_h(h_k), \qquad \mathcal{F}_k = \mathcal{F}' + \Delta\mathcal{F}_k, \qquad p_k = Decoder(\mathcal{F}_k), \quad k = 1, \ldots, D. \tag{9}$$

Each $\mathcal{F}_k$ is retained for subsequent inheritance and h_k is passed to the next step. Historical retrieval preserves accumulated adversarial cues, whereas recurrent refinement organizes them along the same order in which patches are switched during deployment. More explanation are shown in Appendix B .

3.4 Spatio-Temporal Degradation Augmentation

Physical capture changes the appearance and temporal presentation of the displayed patch. We therefore apply degradation to the selected patch stream $\{p_d\}_{d=1}^{D}$ before patch injection. We sample $\xi = (\xi_{\text{tmp}}, \xi_{\text{spa}}) \sim A_{\text{phy}}$ to determine the activation and parameters of each operator. The complete transformation is written in one line:

$$p_{\text{tmp}} = \tau_{\text{fps}}^{a_{\text{fps}}}(\tau_{\text{trail}}^{a_{\text{trail}}}(p)), \qquad \hat{p} = \tau_{\text{res}}^{a_{\text{res}}}\left(\tau_{\text{blur}}^{a_{\text{blur}}}\left(\tau_{\text{glare}}^{a_{\text{glare}}}(p_{\text{tmp}})\right)\right). \tag{10}$$

The first stage applies inter-frame trailing and frame-rate mismatch to the patch stream, whereas the second applies glare, motion blur, and resolution variation. When all $a = 0$, $\widetilde{p}_t = p_{k(t)}$ and the formulation reduces to the digital attack. No additional degradation is introduced: the transformed attacked frame is optimized by $\mathcal{L}_{total}$, while tracker parameters θ remain fixed and S_t evolves normally. Detailed operator definitions are provided in Appendix C .

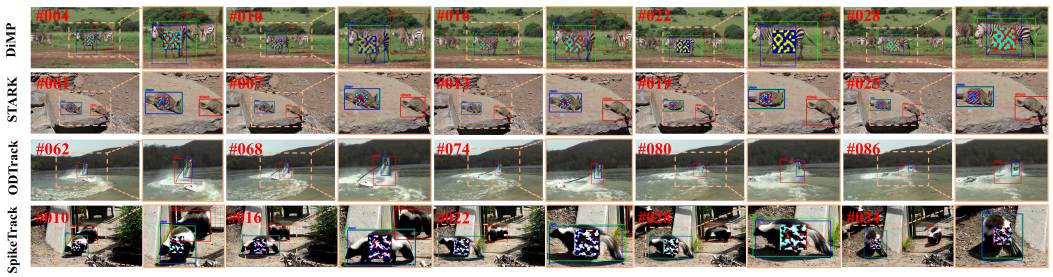


Figure 3: Qualitative comparison on four SOTA online trackers. Green, blue, and red boxes denote ground truth, clean tracking, and the proposed dynamic patch attack, respectively.

4 Experiments

4.1 Experimental Setup

Datasets and Evaluation Metrics We evaluate our method on four widely used tracking benchmarks: OTB100 (Wu et al., 2015), UAV123 (Mueller et al., 2016), GOT-10k (Huang et al., 2019), and LaSOT (Fan et al., 2021), covering conventional, aerial, general, and long-term tracking scenarios. We report AUC as the primary metric, together with AO and SR where applicable. All methods are evaluated under the same experimental protocol for fair comparison. More details are in Appendix D .

Victimized Trackers and Comparison Attackers We consider four representative online trackers, DiMP, STARK, ODTrack, and SpikeTrack, as victim models. We compare against five state-of-the-art adversarial attacks, including MTD, UEN, BAPGNet, CMS, and PANC.

Implementation Details Unless otherwise specified, the patch-to-target scale ratio is set to $\psi = 0.30$, with $N = 5$ patches and a switching interval of 2 frames. For the window-based switching constraint loss, we set $L = 2$, $\delta = 0.03$, and $\lambda_{\text{switch}} = 0.30$, with a weight of 0.80 for the transition from patch 4 to patch 0. All experiments are implemented in PyTorch and run on an NVIDIA TITAN RTX GPU.

Table 1: Performance comparison on different SOTA online trackers (lower is stronger).

Dataset		GOT10k			LaSOT			OTB	UAV
Tracker	Method \ Metric	AO	SR0.5	SR0.75	AUC	P_{norm}	P	AUC	AUC
DiMP (CVPR19)	No attack	50.1	55.7	15.6	56.9	-	-	68.4	65.3
	MTD (AAAI21)	43.6	46.3	13.2	34.2	47.7	39.2	45.8	37.5
	UEN (AAAI21)	46.1	50.3	13.9	34.8	49.9	40.7	46.3	40.7
	BAPGNet (RS23)	51.2	57.9	16.1	37.9	60.6	50.3	57.6	44.4
	CMS (AAAI25)	47.5	52.3	13.9	36.3	54.1	43.4	49.5	42.9
	PANC (TIFS25)	49.2	54.3	15.2	33.4	44.2	35.9	54.5	41.1
	Ours	38.3	37.6	10.5	31.4	39.2	31.4	36.2	30.2
STARK (AAAI21)	No attack	73.8	85.6	65.8	67.1	77.0	72.2	69.1	68.5
	MTD (AAAI21)	55.6	61.4	38.4	42.9	46.4	38.5	49.8	56.9
	UEN (AAAI21)	47.7	49.7	30.8	36.5	36.2	30.0	42.1	49.4
	BAPGNet (RS23)	72.4	83.9	62.2	60.5	72.5	62.5	65.6	67.1
	CMS (AAAI25)	57.2	64.8	36.4	44.3	50.7	40.0	50.4	56.2
	PANC (TIFS25)	58.6	67.5	49.0	46.4	46.6	43.8	59.8	58.5
	Ours	39.9	38.6	16.3	29.3	26.2	21.9	36.2	42.3
ODTrack (AAAI25)	No attack	82.5	93.6	82.9	73.2	83.2	80.6	72.3	-
	MTD (AAAI21)	78.2	90.4	72.9	64.1	72.7	67.1	66.0	62.7
	UEN (AAAI21)	80.9	92.5	78.5	65.8	74.3	69.4	68.0	63.9
	BAPGNet (RS23)	81.5	92.5	79.8	65.5	73.8	69.3	69.1	64.2
	CMS (AAAI25)	81.3	93.2	79.4	64.9	73.3	68.5	69.4	64.8
	PANC (TIFS25)	84.8	94.3	85.3	64.6	72.9	68.5	67.0	62.2
	Ours	75.9	88.7	67.5	61.6	70.0	63.2	63.1	62.4
SpikeTrack (CVPR26)	No attack	79.7	92.5	77.6	65.1	76.4	69.6	67.1	66.3
	MTD (AAAI21)	71.8	82.3	64.5	55.4	63.7	56.9	60.3	66.1
	UEN (AAAI21)	70.0	79.7	63.2	54.9	62.5	55.9	61.1	63.2
	BAPGNet (RS23)	77.9	90.3	73.8	68.2	83.0	75.0	74.1	72.2
	CMS (AAAI25)	69.6	80.1	58.3	54.6	64.8	56.2	61.9	65.8
	PANC (TIFS25)	77.3	90.5	73.9	61.8	72.6	65.1	65.6	64.7
	Ours	68.9	78.6	59.8	53.5	61.1	54.2	60.6	64.6

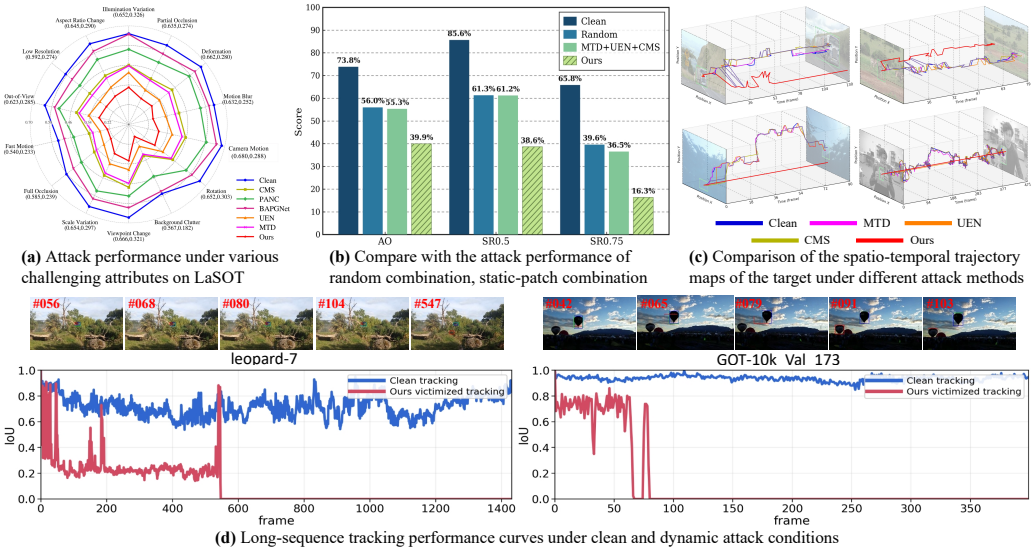
Table 2: Performance comparison on different SOTA offline trackers (lower is stronger).

Dataset		GOT10k			LaSOT			OTB	UAV
Tracker	Method \ Metric	AO	SR0.5	SR0.75	AUC	P_{norm}	P	AUC	AUC
SiamFC++ (AAAI20)	No attack	73.4	85.9	61.1	53.3	57.1	53.9	58.2	57.1
	MTD (AAAI21)	54.3	55.9	33.1	35.1	38.7	32.5	57.4	48.8
	UEN (AAAI21)	57.6	65.2	42.3	29.2	29.3	25.4	54.5	54.1
	BAPGNet (RS23)	65.5	78.1	42.9	47.3	70.3	55.1	67.5	60.6
	CMS (AAAI25)	51.5	57.9	28.8	29.1	36.0	27.5	54.2	51.5
	PANC (TIFS25)	72.7	84.9	66.3	42.8	49.2	42.1	60.4	54.8
	Ours	48.7	52.9	29.4	25.0	26.6	20.6	46.5	48.6
AsymTrack (AAAI25)	No attack	78.2	89.6	74.5	61.1	63.3	61.6	63.8	65.6
	MTD (AAAI21)	72.1	81.0	65.3	56.4	58.3	55.9	57.1	63.2
	UEN (AAAI21)	67.6	75.2	59.0	52.3	50.0	53.7	56.2	62.2
	BAPGNet (RS23)	76.6	87.0	71.7	64.0	73.8	66.3	68.5	69.0
	CMS (AAAI25)	69.5	78.8	61.2	54.3	60.9	53.2	58.8	63.6
	PANC (TIFS25)	73.6	85.9	69.5	58.5	65.7	57.8	60.5	61.4
	Ours	67.0	75.3	59.0	51.2	52.8	49.5	55.2	61.3

4.2 Comparison Experiments and Generalization Evaluation

Fig. 3 illustrates the advantage of DyPatch. While clean tracking and competing attacks generally preserve partial alignment with the target, DyPatch causes the predicted boxes to progressively

deviate and eventually drift toward the background. This persistent error accumulation indicates that patch switching prevents adaptation to a fixed adversarial appearance, while adversarial-prior inheritance propagates misleading information across frames. As shown in Tab. 1, DyPatch achieves the lowest tracking scores in 28 of 32 comparisons and ranks first across all metrics on DiMP and STARK. For example, it reduces GOT-10k AO to 38.3 and 39.9 on the two trackers, outperforming the strongest baselines by 5.3 and 7.8 points. The gains are smaller on ODTrack and SpikeTrack, because their architecture-specific update mechanisms partially absorb abrupt patch changes, while shorter sequences provide less time for adversarial priors to accumulate. To further demonstrate the effectiveness of DyPatch, we use DyPatch to attack the SOTA offline SiamFC++ and AsymTrack trackers. In Tab. 2, DyPatch achieves the best result in 12 of 16 metrics and ties the best result in one additional metric, including seven of eight metrics on SiamFC++. It reduces the GOT-10k AO to 48.7 and 67.0 on SiamFC++ and AsymTrack, respectively. More results are shown in Appendix E .



(a) Attack performance under various challenging attributes on LaSOT

(b) Compare with the attack performance of random combination, static-patch combination

(c) Comparison of the spatio-temporal trajectory maps of the target under different attack methods

(d) Long-sequence tracking performance curves under clean and dynamic attack conditions

Figure 4: Different dynamic patch attack performance.

4.3 Dynamic Patch Attack Performance

Attack Results in the Challenge Scenarios Fig. 4(a) compares attack performance across 14 challenging attributes on LaSOT, where lower values indicate stronger attacks. DyPatch produces the innermost curve across all attributes, with particularly clear advantages under partial occlusion, full occlusion, and fast motion. These conditions frequently change target appearance and weaken static patches, whereas temporal switching prevents adaptation to a fixed pattern. Combined with spatio-temporal degradation augmentation, DyPatch therefore maintains effective interference under appearance, motion, illumination, and background variations.

Dynamic Combined Attack Experiment Fig. 4(b) compares different multi-patch strategies on STARK. Random combination and the direct combination of MTD, UEN, and CMS obtain nearly identical results, indicating that simply stacking multiple patches provides limited additional benefit. In contrast, DyPatch outperforms the combined baseline by 15.4, 22.6, and 20.2 points, respectively. These results show that the improvement arises from temporal switching and adversarial prior inheritance rather than patch quantity alone. Fig. 4(c) interestingly shows that DyPatch can effectively hijack the spatio-temporal trajectory of the target, much better than other attack methods.

Full-Video Attack Performance Fig. 4(d) evaluates whether DyPatch remains effective throughout complete video sequences. While clean trackers maintain relatively stable IoU, DyPatch causes either rapid collapse or progressive degradation, eventually reducing IoU to nearly zero in all four examples. The immediate failure on leopard-7 shows that the attack can produce strong short-term error accumulation. This sustained degradation supports the role of adversarial-prior inheritance in continuously corrupting online updates. More results are shown in Appendix F .

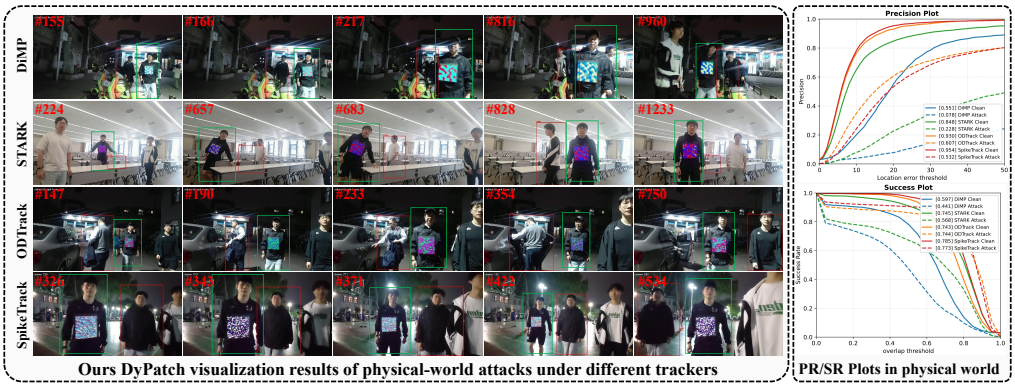


Figure 5: Physical-world tracking results in various real scenarios.

4.4 Physical Attack Experiments

We deploy dynamic patches on a wearable display that switches every 0.5s in physical wolrd. In Fig. 5, DyPatch consistently reduces both PR and SR across all four trackers, demonstrating that the learned dynamic patches remain effective after physical display and image-capture distortions. Fig. 5 also shows that attacked predictions progressively separate from the ground-truth target and eventually drift toward nearby pedestrians or background regions. This behavior is observed across DiMP, STARK, ODTrack, and SpikeTrack under indoor, outdoor, and varying illumination conditions. The persistent failures indicate that cyclic patch switching continues to disrupt feature matching and online adaptation in real scenes.

Table 3: Hyperparameter analysis of patch scale, patch count and switching window on the DiMP tracker.

Factor	Setting	AO	SR0.5	SR0.75
Patch scale s	$\psi = 0.20$	46.0	49.9	13.3
	$\psi = 0.25$	41.7	43.2	12.1
	$\psi = 0.30$	38.4	38.4	10.4
	$\psi = 0.35$	36.3	35.6	10.6
	$\psi = 0.40$	34.5	33.3	10.0
Patch count D	$D = 3$	46.4	50.4	13.8
	$D = 5$	38.4	38.4	10.4
	$D = 7$	42.1	44.1	12.3
Switch window L	$L = 1$	46.7	50.8	13.8
	$L = 2$	38.4	38.4	10.4
	$L = 4$	47.0	51.4	13.7

4.5 Analysis on Hyperparameter Settings and Ablation Studies

Tab. 3 analyzes the patch scale s, patch count D, and switching window L. Increasing s from 0.20 to 0.40 reduces AO from 46.0 to 34.5, but a larger patch also weakens visual concealment. Hence, we use $\psi = 0.30$ as a practical balance. $K = 5$ achieves the best overall performance, as too few patches provide insufficient temporal diversity while too many introduce redundant switching patterns in DiMP. Similarly, $L = 2$ outperforms $L = 1$ and $L = 4$, balancing sensitivity to frame fluctuations with preservation of abrupt temporal interference. More experiments detailed analysis, please refer to the Appendix H & I . Tab. 4 evaluates the $\mathcal{R}_e(\cdot)$, $\mathcal{L}_{tsa}$, $\mathcal{C}u(\cdot)$, and $\mathcal{L}_{atk}$. The complete model achieves the lowest AO, $\text{SR}_{0.5}$, and $\text{SR}_{0.75}$ of 38.3, 37.6, and 10.5, respectively. Remove the $\mathcal{R}_e(\cdot)$ increases these metrics to 43.2, 43.9, and 22.3. Removing the $\mathcal{L}_{tsa}$ or $\mathcal{C}u(\cdot)$ causes further degradation, while the $\mathcal{L}_{atk}$ provides a smaller but consistent improvement.

Table 4: Ablation of core components in GOT-10k.

No.	$\mathcal{R}_e(\cdot)$	$\mathcal{L}_{tsa}$	$\mathcal{C}u(\cdot)$	$\mathcal{L}_{atk}$	AO	$\text{SR}_{0.5}$	$\text{SR}_{0.75}$
I	×	✓	✓	✓	43.2	43.9	22.3
II	✓	×	✓	✓	44.9	46.8	23.2
III	✓	✓	×	✓	46.4	50.5	13.7
IV	✓	✓	✓	×	39.3	39.2	10.9
V	✓	✓	✓	✓	**38.3**	**37.6**	**10.5**

5 Conclusion

We introduce DyPatch to exploit the temporal dependencies created by continuous model updates in online trackers. Instead of repeatedly applying a fixed patch, DyPatch jointly optimizes a multi-patch sequence through temporal switching, adversarial-prior inheritance, and spatio-temporal degradation augmentation, producing persistent interference across consecutive frames. Experiments show that DyPatch effectively attacks online and offline trackers in both digital and physical domain.

AI USE STATEMENT

Generative AI tools were used for language polishing, grammar checking and auxiliary software troubleshooting. All AI suggestions were reviewed by the authors. AI was not used for research idea generation, method/experiment design, coding, data generation/processing, or result interpretation. The authors are fully responsible for all contents of this work.

REFERENCES

Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6182–6191, 2019.

Wenrui Cai, Qingjie Liu, and Yunhong Wang. Hiptrack: Visual tracking with historical prompts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19258–19267, 2024.

Wenrui Cai, Qingjie Liu, and Yunhong Wang. Spmtrack: Spatio-temporal parameter-efficient fine-tuning with mixture of experts for scalable visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16871–16881, 2025.

Li Ding, Yongwei Wang, Kaiwen Yuan, Minyang Jiang, Ping Wang, Hua Huang, and Z Jane Wang. Towards universal physical attacks on single object tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 1236–1245, 2021.

Zhaodong Ding, Chenglong Li, Shengqing Miao, and Jin Tang. Template-based uncertainty multi-modal fusion network for rgbt tracking. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 909–917, 2025a.

Zhaodong Ding, Chenglong Li, Tao Wang, and Futian Wang. Quality-aware spatio-temporal transformer network for rgbt tracking. *IEEE Transactions on Image Processing*, 34:7845–7858, 2025b.

Heng Fan, Hexin Bai, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Harshit, Mingzhen Huang, Juehuan Liu, et al. Lasot: A high-quality large-scale single object tracking benchmark. *International Journal of Computer Vision*, 129:439–461, 2021.

Changhong Fu, Sihang Li, Xinnan Yuan, Junjie Ye, Ziang Cao, and Fangqiang Ding. Ad2attack: Adaptive adversarial attack on real-time uav tracking. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 5893–5899, 2022.

Peng Gao, Long Xu, Wen-Jia Tang, Fei Wang, Hamido Fujita, Hanan Aljuaid, and Ru-Yue Yuan. Towards patch-based noise compression for adversarial attack against transformer-based visual tracking. *IEEE Transactions on Information Forensics and Security*, 21:854–869, 2025.

Mingzhe Guo, Weiping Tan, Wenyu Ran, Liping Jing, and Zhipeng Zhang. Dreamtrack: Dreaming the future for multimodal visual object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7201–7210, 2025.

Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43 (5):1562–1577, 2019.

Xingsen Huang, Yuqing Huang, Hongpeng Wang, Yaowei Wang, and Xin Li. Saliency-aware adversarial attacks on visual trackers. *IEEE Transactions on Multimedia*, 28:7305–7317, 2026.

Erkenbieke Jia, Yuelei Xu, Zhaoxiang Zhang, Chengyang Tao, and Tian Hui. Bilateral adversarial patch generating network for the object tracking algorithm. *Remote Sensing*, 15(14):3670, 2023.

Shuai Jia, Yibing Song, Chao Ma, and Xiaokang Yang. 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*, pp. 6705–6714, 2021.

Shuai Jia, Chao Ma, Yibing Song, Xiaokang Yang, and Ming-Hsuan Yang. Robust deep object tracking against adversarial attacks. *International Journal of Computer Vision*, 133(3):1238–1257, 2025.

Jingjing Li, Hanlin Qin, Yue Yu, Shuai Yuan, Xupei Zhang, Juliu Li, and Steven Landgraf. Dsrtrack: Disturbance-aware state recovery and volatility regulation for rgb-t tracking. *Information Fusion*, pp. 104803, 2026.

Shengtao Lou, Buyu Liu, Jun Bao, Jiajun Ding, and Jun Yu. Follow-me: Deceiving trackers with fabricated paths. In *Proceedings of the ACM International Conference on Multimedia*, pp. 8808–8818, 2023.

Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *Proceedings of the European Conference on Computer Vision*, volume 7, pp. 445–461, 2016.

Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015.

Xinyu Xiang, Qinglong Yan, Hao Zhang, Jianfeng Ding, Han Xu, Zhongyuan Wang, and Jiayi Ma. Cross-modal stealth: A coarse-to-fine attack framework for rgb-t tracker. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 8620–8627, 2025a.

Xinyu Xiang, Qinglong Yan, Hao Zhang, and Jiayi Ma. Acattack: Adaptive cross attacking rgb-t tracker via multi-modal response decoupling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 22099–22108, June 2025b.

Xinyu Xiang, Xuying Wu, Shengxiang Li, Qinglong Yan, Tong Zou, Hao Zhang, and Jiayi Ma. Adversarial perturbation for rgb-t tracking via intra-modal excavation and cross-modal collusion. *Information Fusion*, pp. 104183, 2026a.

Xinyu Xiang, Qinglong Yan, Shengxiang Li, Linfeng Tang, Zizhuo Li, Hao Zhang, and Jiayi Ma. From single-modal to multi-modal: A survey and perspective on adversarial attacks in multi-modal visual systems. *Information Fusion*, pp. 104718, 2026b.

Shaoyuan Xie, Mohamad Habib Fakih, Junchi Lu, Fayzah Alshammari, Ningfei Wang, Takami Sato, Halima Bouzidi, Mohammad Abdullah Al Faruque, and Qi Alfred Chen. Flytrap: Physical distance-pulling attack towards camera-based autonomous target tracking systems. In *Proceedings of the Network and Distributed System Security Symposium*, 2026.

Yinda Xu, Zeyu Wang, Zuoxin Li, Ye Yuan, and Gang Yu. Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 12549–12556, 2020.

CHEN Xuesong, FU Canmiao, Feng ZHENG, et al. A unified multi-scenario attacking network for visual object tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 1097–1104, 2021.

Bin Yan, Dong Wang, Huchuan Lu, and Xiaoyun Yang. Cooling-shrinking attack: Blinding the tracker with imperceptible noises. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 990–999, 2020.

Bin Yan, Houwen Peng, Jianlong Fu, Dong Wang, and Huchuan Lu. Learning spatio-temporal transformer for visual tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10448–10457, 2021.

Qiuyang Zhang, Jiujun Cheng, Qichao Mao, Cong Liu, Yu Fang, Yuhong Li, Mengying Ge, and Shangce Gao. Spiketrack: A spike-driven framework for efficient visual tracking. *arXiv preprint arXiv:2602.23963*, 2026.

Shaochuan Zhao, Tianyang Xu, Xiao-Jun Wu, and Josef Kittler. Pluggable attack for visual object tracking. *IEEE Transactions on Information Forensics and Security*, 19:1227–1240, 2023.

Shaochuan Zhao, Tianyang Xu, Yong Zhou, Zhangyong Tang, Xiao-Jun Wu, and Josef Kittler. Attack intensity is target-related: Exploration of sparse adversarial attack against visual object trackers. *International Journal of Computer Vision*, 134(7):327, 2026.

Yaozong Zheng, Bineng Zhong, Qihua Liang, Zhiyi Mo, Shengping Zhang, and Xianxian Li. Odtrack: Online dense temporal token learning for visual tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 7588–7596, 2024.

Ze Zhou, Yinghui Sun, Quansen Sun, Chaobo Li, and Zhenwen Ren. Only once attack: Fooling the tracker with adversarial template. *IEEE Transactions on Circuits and Systems for Video Technology*, 33(7):3173–3184, 2023.

Jiawen Zhu, Huayi Tang, Xin Chen, Xinying Wang, Dong Wang, and Huchuan Lu. Two-stream beats one-stream: Asymmetric siamese network for efficient visual tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 10959–10967, 2025.

A Temporal Switching Implementation

The patch selector $k(t)$ in Section 3.1 determines which p_d is displayed at frame t. We collect the transition frames as

$$\mathcal{S} = \{t \in \{2, \dots, T\} \mid k(t) \neq k(t-1)\}. \tag{11}$$

Each $t \in \mathcal{S}$ is the first frame rendered with a new patch. Following Section 3.2, q_t denotes attack quality, for which a larger value represents a stronger attack. The pre-switch reference is computed over the preceding L frames:

$$\Omega_t^- = \{t - L, \dots, t - 1\}. \tag{12}$$

$$\bar{q}_t = \frac{1}{L} \sum_{l \in \Omega_t^-} q_l. \tag{13}$$

We use $L = 2$. Averaging the frames immediately before a transition provides a local reference for the current tracker state and prevents the constraint from being determined by a single pre-switch prediction.

Ordinary transitions and the wrap-around transition use different margins and weights:

$$(\delta_t, \lambda_{switch}) = \begin{cases} (0.06, 0.80), & k(t-1) = D \text{ and } k(t) = 1, \\ (0.03, 0.30), & \text{otherwise.} \end{cases} \tag{14}$$

The cyclic transition receives the stronger constraint because it closes the deployment sequence and links p_D back to p_1. The complete switching loss is

$$\mathcal{L}_{tsa} = \lambda_{switch} \sum_{t \in \mathcal{X}} \max\big(0, \bar{q}_t - (q_t - \delta_t)\big). \tag{15}$$

The hinge term is active when the first post-switch attack quality does not exceed its pre-switch reference by the required margin. It therefore rewards a transition only when the new patch strengthens the attack, rather than merely producing an abrupt visual change. As in Eq. 2, $\mathcal{L}_{tsa}$ is jointly optimized with $\mathcal{L}_{atk}$ through the coefficient λ_{switch}.

B Architectural Details of Adversarial Prior Inheritance

The generator produces the ordered patch sequence recursively. We initialize a learnable feature $\mathcal{F}_1$, decode the first patch as $p_1 = Decoder(\mathcal{F}_1)$, and set the recurrent state to $h_1 = 0$. When generating $\mathcal{F}_k$ for $k = 2, \dots, D$, the immediately preceding feature $\mathcal{F}_{k-1}$ serves as the current input, while earlier features form the historical feature bank used in the main text:

$$\mathcal{M}_b^{k-1} = \begin{cases} \emptyset, & k = 2, \\ \{\mathcal{F}_1, \dots, \mathcal{F}_{k-2}\}, & k \geq 3. \end{cases} \tag{16}$$

A generated feature is retained and becomes available to the bank after it is no longer the current feature, ensuring that retrieval uses only preceding adversarial states.

For a nonempty history bank, global average pooling converts each historical feature into a compact token. These tokens are stacked in their generation order and projected into keys and values, while the current feature supplies the query:

$$\mathcal{H}_{k-1} = Stack[GAP(\mathcal{F}_1), \dots, GAP(\mathcal{F}_{k-2})], \tag{17}$$

$$\mathbf{Q} = g_q(\mathcal{F}_{k-1}), \tag{18}$$

$$\mathbf{K} = g_k(\mathcal{H}_{k-1}), \tag{19}$$

$$\mathbf{V} = g_v(\mathcal{H}_{k-1}). \tag{20}$$

The attention weights select historical components that are most relevant to the current adversarial feature:

$$A_{k-1} = Softmax\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right), \tag{21}$$

$$\mathcal{F}_{\text{attn}} = \mathcal{F}_{k-1} + g_o(A_{k-1}\mathbf{V}). \tag{22}$$

If $\mathcal{M}_b^{k-1}$ is empty, attention retrieval is skipped and $\mathcal{F}_{\text{attn}} = \mathcal{F}_{k-1}$. This boundary case allows the recursive generator to start without introducing an artificial history token.

The retrieved feature is processed by a lightweight adapter consisting of a 3×3 convolution, batch normalization, ReLU, and a residual block. Its output forms the first residual update:

$$\Delta\mathcal{F}_{k-1} = ResBlock(ReLU(BN(Conv_{3\times3}(\mathcal{F}_{\text{attn}})))) \,. \tag{23}$$

$$\mathcal{F}' = \mathcal{F}_{k-1} + \Delta\mathcal{F}_{k-1}. \tag{24}$$

The residual path preserves the preceding adversarial feature, while $\Delta\mathcal{F}_{k-1}$ injects the historical information selected by cross-attention.

For recurrent refinement, the adapted feature is combined with the projected preceding state:

$$X_k = Proj(\text{concat}(\mathcal{F}', W_h(h_{k-1}))) \,, \tag{25}$$

where $\text{concat}(\cdot)$ denotes channel-wise concatenation and $Proj$ is a 1×1 projection. The ConvGRU then updates its gates and hidden state as

$$z_k = \sigma(W_z * \text{concat}(X_k, h_{k-1})) \,. \tag{26}$$

$$r_k = \sigma(W_r * \text{concat}(X_k, h_{k-1})) \,. \tag{27}$$

$$\widetilde{h}_k = \tanh(W_c * \text{concat}(X_k, r_k \odot h_{k-1})) \,. \tag{28}$$

$$h_k = (1 - z_k) \odot h_{k-1} + z_k \odot \widetilde{h}_k. \tag{29}$$

Here, $*$ denotes convolution. The updated state is projected into the second residual and used to obtain the next patch feature:

$$\Delta\mathcal{F}_k = W_h(h_k). \tag{30}$$

$$\mathcal{F}_k = \mathcal{F}' + \Delta\mathcal{F}_k. \tag{31}$$

$$p_k = Decoder(\mathcal{F}_k), \qquad k = 2, \ldots, D. \tag{32}$$

Unlike $\Delta\mathcal{F}_{k-1}$, which retrieves explicit information from earlier patch features, $\Delta\mathcal{F}_k$ carries the recurrent state accumulated along the generation order. Their combination makes each patch inherit the existing adversarial tendency while retaining the capacity to introduce a new perturbation for the next switching stage.

C Spatio-Temporal Degradation Details

Physical degradation is applied to the selected patch before it is injected into the video frame. For the patch stream $\mathbf{p} = \{p_{k(t)}\}_{t=1}^{T}$, we initialize the degradation chain as

$$p_t^{(0)} = p_{k(t)}. \tag{33}$$

For each operator τ_j, the sampled configuration $\xi_j = (a_j, \eta_j)$ contains an activation variable $a_j \in \{0, 1\}$ and its transformation parameters η_j. Its switched form is

$$\tau_j^{a_j}(u; \eta_j) = \begin{cases} \tau_j(u; \eta_j), & a_j = 1, \\ u, & a_j = 0. \end{cases} \tag{34}$$

The joint sample $\xi = \{\xi_j\}_j \sim \mathcal{A}_{\text{phy}}$ determines which degradations are active and their strengths. When all $a_j = 0$, the output remains the original selected patch.

Temporal Degradation Inter-frame trailing models the mixture of consecutive displayed patches caused by display persistence and camera exposure. With $\alpha \sim \mathcal{D}_{\text{trail}}$, the active operator is

$$\tau_{\text{trail}}(p_t^{(0)}; \alpha) = \begin{cases} p_1^{(0)}, & t = 1, \\ \alpha p_t^{(0)} + (1 - \alpha) p_{t-1}^{(0)}, & t > 1. \end{cases} \tag{35}$$

The switched trailing output is

$$p_t^{(1)} = \tau_{\text{trail}}^{a_{\text{trail}}}(p_t^{(0)}; \alpha). \tag{36}$$

Thus, the transition between consecutive dynamic patches is exposed to the same temporal integration that occurs when a displayed pattern is recorded by a camera.

Frame-rate mismatch is then simulated by an order-preserving remapping of the patch sequence. For $\nu \sim \mathcal{D}_{\text{fps}}$, the source index is

$$\iota_\nu(t) = clip(\lfloor \nu t \rfloor, 1, T)\,. \tag{37}$$

The resulting temporal output is

$$\begin{aligned} p_t^{(2)} &= \tau_{\text{fps}}^{a_{\text{fps}}}(p_t^{(1)}; \nu), \\ \tau_{\text{fps}}(p_t^{(1)}; \nu) &= p_{\iota_\nu(t)}^{(1)}. \end{aligned} \tag{38}$$

Values of $\nu < 1$ repeat some patch observations, whereas $\nu > 1$ skips them. Since $\iota_\nu(t)$ is monotonic, the effective presentation rate changes without shuffling the learned patch order.

Spatial Degradation The spatial branch follows the temporal branch and applies glare, motion blur, and resolution variation to the patch. Given a patch-sized highlight mask G_t and $g \sim \mathcal{D}_{\text{glare}}$, glare is modeled as

$$\begin{aligned} p_t^{(3)} &= \tau_{\text{glare}}^{a_{\text{glare}}}(p_t^{(2)}; g, G_t), \\ \tau_{\text{glare}}(u; g, G_t) &= clip(u + gG_t, 0, 1). \end{aligned} \tag{39}$$

For a motion-blur kernel $K_{\kappa,\varphi}$, where its extent κ and direction φ are sampled from $\mathcal{D}_{\text{blur}}$, we obtain

$$\begin{aligned} p_t^{(4)} &= \tau_{\text{blur}}^{a_{\text{blur}}}(p_t^{(3)}; \kappa, \varphi), \\ \tau_{\text{blur}}(u; \kappa, \varphi) &= K_{\kappa,\varphi} * u. \end{aligned} \tag{40}$$

Finally, with $\rho \sim \mathcal{D}_{\text{res}}$, resolution degradation downsamples the patch and restores it to its deployment size:

$$\begin{aligned} p_t^{(5)} &= \tau_{\text{res}}^{a_{\text{res}}}(p_t^{(4)}; \rho), \\ \tau_{\text{res}}(u; \rho) &= \mathcal{R}_\uparrow(\mathcal{R}_\downarrow(u; \rho); \rho)\,. \end{aligned} \tag{41}$$

Here, $*$ denotes convolution, while $\mathcal{R}_\downarrow$ and $\mathcal{R}_\uparrow$ denote downsampling and restoration. The final degraded patch and the corresponding tracker input are

$$\widetilde{p}_t = p_t^{(5)}. \tag{42}$$

$$\widetilde{x}_t^{\text{adv}} = x_t \odot (1 - M_p) + \widetilde{p}_t \odot M_p. \tag{43}$$

During physical-aware optimization, $\widetilde{x}_t^{\text{adv}}$ replaces x_t^{adv} in $\mathcal{L}_{total} = \mathcal{L}_{atk} + \mathcal{L}_{tsa}$. The augmentation introduces no additional degradation loss: the tracker parameters θ remain fixed, while its online state S_t continues to evolve through Φ on the transformed attacked frames.

D Dataset Details

OTB100 OTB100 is a conventional single-object tracking benchmark containing 100 fully annotated video sequences. It covers common challenges such as occlusion, scale variation, deformation, fast motion, background clutter, and illumination change. We report the area under the success curve (AUC), which averages the proportion of frames whose predicted box exceeds different overlap thresholds. OTB100 provides a compact evaluation of whether an attack remains effective across common appearance and motion changes without relying on very long sequences.

UAV123 UAV123 contains 123 sequences captured from low-altitude UAV platforms. The benchmark frequently contains camera motion, small targets, low spatial resolution, fast motion, and background clutter. We use success AUC as the evaluation metric. Because the patch scale is proportional to the target box, UAV123 also tests whether the learned adversarial pattern remains effective when the rendered patch occupies only a small number of pixels after target cropping and backbone downsampling.

GOT-10k GOT-10k is a large-scale generic tracking benchmark with more than 10,000 video segments and broad diversity in object categories and motion patterns. Its one-shot protocol separates the object classes used for training and testing, making it suitable for evaluating attack generalization to unseen target categories. We report average overlap (AO) together with $SR_{0.5}$ and $SR_{0.75}$, which measure the proportions of frames whose overlap exceeds 0.5 and 0.75, respectively. The two success thresholds distinguish general localization degradation from failures that specifically destroy high-precision tracking.

LaSOT LaSOT is a large-scale, densely annotated benchmark composed of 1,400 sequences with an average length exceeding 2,500 frames. Its long videos contain target disappearance, reappearance, occlusion, and substantial appearance variation. We report success AUC, normalized precision, and precision. The long temporal horizon permits many patch cycles and tracker updates, making LaSOT particularly informative for determining whether an attack is gradually absorbed by the tracker or continues to accumulate interference over time.

MOT17 MOT17 contains pedestrian tracking sequences recorded from different viewpoints and scene conditions. We follow the half-train/half-validation protocol and evaluate ByteTrack on the latter halves of the seven training sequences. This split provides ground-truth trajectories for measuring how transferred patches affect both pedestrian localization and identity continuity under a range of crowd densities and camera motions.

MOT20 MOT20 focuses on densely crowded pedestrian scenes with frequent occlusion and close interaction between targets. Because an equivalent validation-half annotation is unavailable, we evaluate on the complete training sequences with accessible ground truth. Its high target density makes association particularly sensitive to missing detections and ambiguous target evidence, complementing the more varied MOT17 evaluation.

For both benchmarks, we report MOTA, IDF1, and HOTA. MOTA summarizes detection and localization errors, IDF1 emphasizes identity preservation, and HOTA jointly reflects detection and association quality. Using all three metrics helps distinguish a reduction in per-frame detections from a loss of temporal identity consistency.

E More Comparison Results and Analysis

E.1 Qualitative Results of Attack Comparison and Offline Tracker Attacks

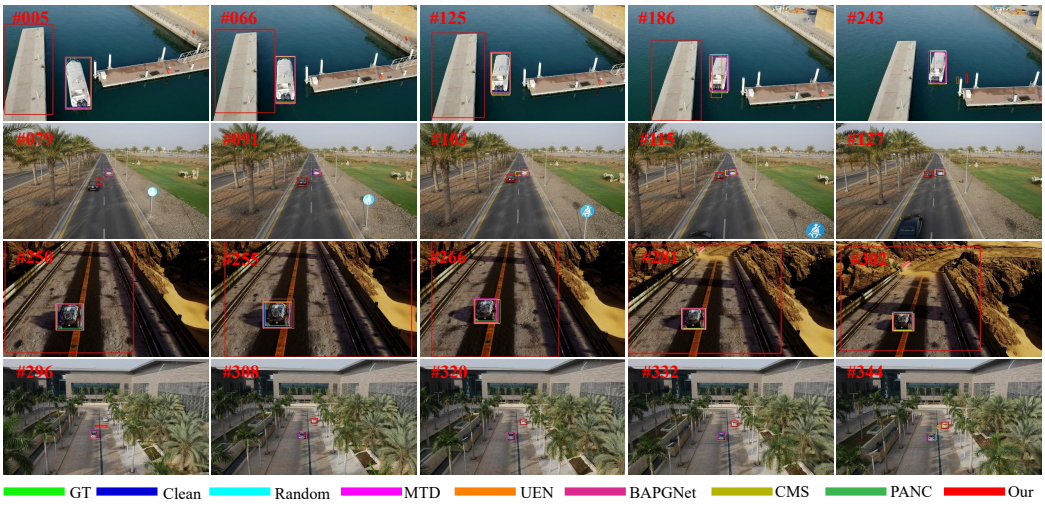


Figure 6: Visualization of tracking deviations under different adversarial methods. The bottom legend maps each bounding-box color to the corresponding clean baseline or attack method.

As illustrated in Fig. 6, the tracking boxes of other comparative methods can still maintain basic alignment with the target in most frames, while the proposed dynamic patch induces significant offset of tracking boxes. Combining quantitative and visual results, the dynamic adversarial patch attack proposed in this work not only achieves state-of-the-art overall performance degradation, but also exhibits sustained and cumulative temporal adversarial effects. By virtue of adversarial prior inheritance and dynamic switching mechanisms, the proposed method precisely disturbs the historical memory learning process of trackers, providing an effective solution for adversarial attacks on prior-enhanced visual trackers.

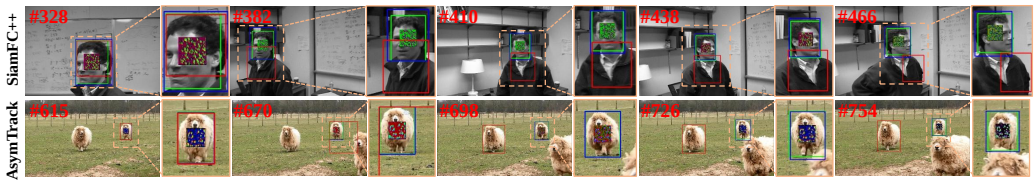


Figure 7: Visualization of tracking deviations under different adversarial methods. The bottom legend maps each bounding-box color to the corresponding clean baseline or attack method.

From the qualitative visualization results in Fig. 7, targets attached with the dynamic patch gradually drift away from predicted bounding boxes across various tracking scenarios, eventually triggering severe tracker drift or complete target loss. Although SiamFC++ and AsymTrack dispense with sophisticated historical memory and temporal modeling modules, the dynamic patch persistently and progressively breaks feature matching between templates and search regions to mislead the localization decisions of trackers continuously. Such observations solidly demonstrate the strong generalization attack performance of our method for tracking algorithms with different architectures.

E.2 Why DyPatch Has a Smaller Effect on SpikeTrack

Architecture-based explanation. The effectiveness of a temporal attack depends on whether an attacked observation affects only the current prediction or is also stored by the tracker and carried into later frames. DiMP updates an online target model from historical samples, while STARK can incorporate recent observations through dynamic templates (Bhat et al., 2019; Yan et al., 2021); both mechanisms provide a relatively direct route for successive patches to accumulate errors. SpikeTrack (Zhang et al., 2026) restricts this route. Its search branch mainly compares the current frame with a cached template memory, and the search observation modifies this persistent memory only when a scheduled, confidence-controlled template update is accepted. Meanwhile, its spike-driven backbone communicates features through thresholded firing responses, so a perturbation must be strong and consistent enough to change these responses before it can propagate through the network. The cached template therefore remains a comparatively stable target reference, while part of the patch-induced variation can be attenuated by spike activation. Together, restricted memory updates and thresholded feature transmission explain why DyPatch produces less temporal error accumulation on SpikeTrack than on DiMP and STARK. They do not make SpikeTrack immune: patches that alter the firing pattern or enter an accepted template update can still affect subsequent predictions.

Dataset Effects The dataset determines how often DyPatch can activate the remaining accumulation path. LaSOT contains long sequences, providing many patch cycles and more opportunities for an attacked observation to enter a template update; DyPatch consequently achieves the strongest attack on all three reported LaSOT metrics for SpikeTrack. On GOT-10k, its diverse one-shot targets increase dependence on template-conditioned matching, and DyPatch is best on AO and $SR_{0.5}$ and second-best on $SR_{0.75}$. The smaller reductions on OTB100 and UAV123 arise under less favorable conditions for accumulation. OTB100 provides fewer long-duration trajectories than LaSOT, while UAV123 frequently contains small, low-resolution targets. Because the patch size is proportional to the target box, the effective patch may occupy only a small number of pixels after cropping and backbone downsampling in these aerial sequences; spike thresholding and stable template retrieval can then suppress part of its signal. Thus, the dataset does not create a separate defense, but controls

the available patch resolution, the number of switching cycles, and the probability that adversarial evidence reaches the persistent template memory.

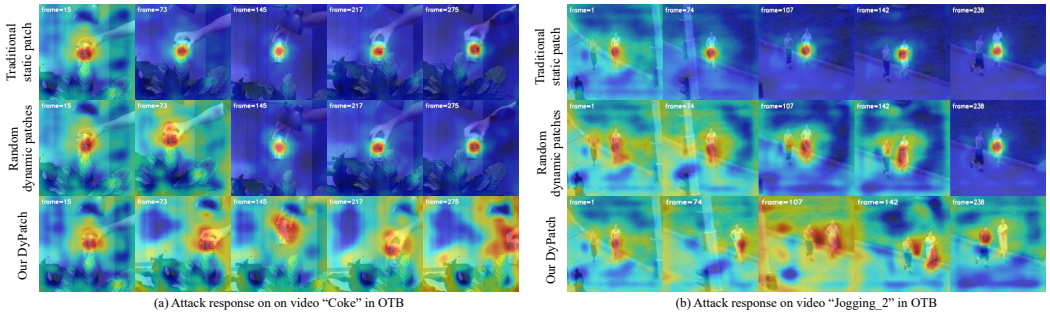
Implications for Tracker Security These observations suggest that limiting writes to persistent tracker states may improve robustness. Stable templates, confidence-aware updates, temporal-consistency checks, and spike-based filtering are possible directions rather than complete defenses. DyPatch still leads five of SpikeTrack's eight digital metrics, ranks second on two others, and reduces physical PR by 32.2 points.

F More Dynamic Patch Attack Performance

Temporal Persistence of Different Patch Strategies As shown in Fig. 4(c), qualitative comparisons on two representative sequences reveal distinct temporal behaviors among the three patch strategies. Under a fixed single patch, the predicted response remains concentrated near the target over most of the sequence. Although the patch changes the local appearance, the same pattern is repeatedly observed and can be incorporated into the target representation maintained by a history-based tracker. The tracker therefore retains a stable reference and suppresses much of the initial disturbance. The uncoordinated combination of MTD, UEN, and CMS produces a larger response displacement immediately after a patch change. However, these patches are generated independently and do not preserve a shared adversarial direction. Their interference is consequently transient: historical target cues re-establish the target response after several frames, and the prediction gradually returns toward the original trajectory.

DyPatch exhibits a different pattern. Adversarial-prior inheritance transfers the misleading feature tendency accumulated by earlier patches into the generation of the next patch, reducing cancellation between successive attack patterns. The switching constraint further requires the post-transition GIoU to remain below its pre-switch reference over a short temporal window. Thus, a patch change is optimized not only to perturb the transition frame but also to preserve the degradation immediately afterward. The two mechanisms make consecutive patches act as connected stages of the same temporal attack, producing a displaced target response that persists across the observed sequence rather than a short-lived error followed by recovery.

The Response Changes of the Tracker Under Different Attacks The response map embodies the tracker's perception and understanding of the target, directly determining tracking robustness and the precision of the predicted bounding box. We extract the response maps of the tracker under different attack methods, as shown in Fig. 8. Our method persistently attacks the tracker across multiple scenarios, and the induced perturbation is difficult to smooth out. In contrast, both static patches and randomly combined dynamic patches fail to sustain the disruption: the response map stabilizes again after a brief period, indicating that the tracker has re-established a confident localization and absorbed the attack into its target representation. Our dynamic patches, however, keep the response map persistently corrupted, preventing the tracker from recovering a sharp peak and thereby sustaining the attack throughout the sequence. This qualitative evidence aligns with our central analysis: only a patch that continuously evolves can keep disturbing the tracker's perception, whereas a stationary signal is gradually assimilated by the state-update process.



(a) Attack response on on video "Coke" in OTB (b) Attack response on video "Jogging_2" in OTB

Figure 8: Response map of different attacks.

G Qualitative and Quantitative Results on Multi-Object Tracking

We transfer the learned patches to ByteTrack by attaching a patch to the center of each target instance under the same cyclic deployment rule. As shown in Fig. 9, qualitative inspection of the two illustrated sequences reveals two recurring outcomes. First, bounding boxes associated with some patched targets disappear in later frames, indicating that the corresponding detection or trajectory is no longer retained. Second, a target that remains localized may receive a different identity after a patch transition. These observations show that an attack trained for single-object localization can also affect the detection and association stages of a multi-object tracking pipeline.



Figure 9: Qualitative results of multi-object tracking before and after being attacked by the digital domain DyPatch.

DyPatch is not explicitly optimized to select or exchange track identities. Its effect follows from repeatedly altering the detection evidence used by ByteTrack across consecutive frames. ByteTrack first associates high-score detections with motion-predicted track boxes and then attempts to recover unmatched tracks using lower-score detections. A patch-induced reduction in detection confidence can therefore remove a valid candidate from association, while a localization shift can reduce its overlap with the predicted track box. Either outcome may interrupt or terminate the existing trajectory, producing the observed box disappearance. If a later detection is initialized as a new trajectory rather than matched back to the lost one, the same pedestrian receives a different identity. The qualitative results therefore connect the reductions in MOTA, IDF1, and HOTA in Tab. 5 to disrupted detection retention and box-level association, without implying a targeted identity attack.

Table 5: Transfer attack performance on multiple object tracking benchmarks. Lower MOTA, IDF1 and HOTA indicate stronger attack effectiveness.

Method	MOT17			MOT20		
	MOTA	IDF1	HOTA	MOTA	IDF1	HOTA
Clean	71.1	74.7	63.2	62.5	64.3	53.1
Random	65.1	68.8	57.2	43.7	44.7	36.3
MTD	64.2	69.7	57.4	43.7	44.2	36.0
UEN	65.7	69.9	57.9	43.3	44.7	36.1
Ours	56.2	62.9	51.4	30.4	33.8	27.5

H More Analysis on Hyperparameter Settings

Patch Dwell Time We first vary the number of consecutive frames for which each patch is displayed while preserving the learned patches and their original order. As shown in Tab. 6, the attack remains

broadly stable over $\Delta \in \{1, 2, 5, 10\}$. On GOT-10k, the AO ranges are only 0.2, 0.5, and 0.9 points for DiMP, STARK, and SpikeTrack, respectively. Their corresponding SR and AUC values also fluctuate within relatively narrow ranges, indicating that the learned patch sequence remains effective under different switching frequencies. ODTrack shows greater sensitivity: compared with $\Delta = 2$, longer dwell times increase its GOT-10k AO by up to 2.2 points and $SR_{0.75}$ by up to 5.0 points. Nevertheless, the default setting achieves the strongest ODTrack attack on all three GOT-10k metrics and on OTB100, while the differences among dwell times on UAV123 remain within 1.0 point.

This stability follows from the sequence-level design of DyPatch. The GIoU objective optimizes every patched observation, while the switching constraint explicitly strengthens the short window following each transition. Varying Δ changes the number and temporal locations of these transitions, but leaves both the learned patch content and the inherited adversarial progression unchanged. In addition, history-based trackers aggregate observations over time and update their target states according to their own update schedules. A moderate shift in the patch boundary therefore does not necessarily coincide with a qualitatively different tracker state, and its effect can be smoothed by the accumulated historical representation. The larger variation on ODTrack further shows that the exact sensitivity remains architecture dependent. We adopt $\Delta = 2$ because it provides frequent transition events and yields a strong overall attack without requiring a switch at every frame.

Table 6: Attack performance under different patch dwell times. GOT-10k reports AO, $SR_{0.5}$, and $SR_{0.75}$; OTB100 and UAV123 report AUC. Lower values indicate stronger attacks. The learned patches and their order remain fixed.

Dataset		GOT10k			OTB	UAV
Tracker	Setting \ Metric	AO	SR0.5	SR0.75	AUC	AUC
DiMP	$\Delta = 2$	38.3	37.6	10.5	36.2	30.2
	$\Delta = 1$	38.4	38.0	10.7	36.8	29.9
	$\Delta = 5$	38.3	38.0	10.6	36.3	30.1
	$\Delta = 10$	38.5	37.9	10.7	36.8	31.9
STARK	$\Delta = 2$	39.9	38.6	16.3	36.2	42.3
	$\Delta = 1$	39.4	38.1	15.6	35.8	41.7
	$\Delta = 5$	39.7	38.5	15.7	35.6	41.8
	$\Delta = 10$	39.5	38.0	15.4	35.6	41.9
ODTrack	$\Delta = 2$	75.9	88.7	67.5	63.1	62.4
	$\Delta = 1$	77.8	90.9	71.9	64.2	62.7
	$\Delta = 5$	78.1	91.0	72.5	64.9	62.3
	$\Delta = 10$	78.1	91.0	72.4	64.4	63.3
SpikeTrack	$\Delta = 2$	68.9	78.6	59.8	60.6	64.6
	$\Delta = 1$	68.0	77.0	58.6	61.1	64.6
	$\Delta = 5$	68.5	77.8	59.4	61.5	64.9
	$\Delta = 10$	68.9	78.1	59.5	62.0	65.1

Patch Deployment Order We next retain the same five learned patches but permute their cyclic deployment order. Tab. 7 shows a consistent advantage for the generation order $0 \to 1 \to 2 \to 3 \to 4$ on ODTrack. Relative to the strongest permuted alternative, the original order lowers GOT-10k AO, $SR_{0.5}$, and $SR_{0.75}$ by 1.9, 2.1, and 3.7 points, respectively, and lowers OTB100 AUC by 1.2 points. The three permutations produce similar degradation to one another but consistently weaker results than the original order on these four metrics. This pattern indicates that the benefit is associated with the learned ordering rather than with one particular random permutation.

The result agrees with the sequential structure of adversarial-prior inheritance. During generation, P_k receives information accumulated through F_{k-1}, the earlier feature bank, and the recurrent state. During optimization, the switching constraint is also evaluated on the transitions induced by this same deployment order, including the cyclic transition from P_{N-1} back to P_0. The original order therefore preserves both the inherited feature progression and the transition pairs optimized during training. Reordering the patches does not remove their individual adversarial patterns, but it replaces

the learned predecessor–successor relationships with unseen transitions. The resulting attack remains effective, yet its temporal coordination is reduced.

UAV123 presents one small exception: the order $2 \to 0 \to 4 \to 1 \to 3$ lowers AUC from 62.4 to 62.1. Targets in UAV123 are frequently small and low resolution, so the target-aligned patch may contain only a limited number of effective pixels after resizing and feature downsampling. Under this condition, small localization and resampling differences can have an effect comparable to the 0.3-point ordering gap. Because this reversal is limited to UAV123 and does not extend to the other metrics, it is more consistent with small-target evaluation variability than with a general advantage of the permuted sequence.

Table 7: Attack performance of different patch deployment orders on ODTrack. GOT-10k reports AO, $SR_{0.5}$, and $SR_{0.75}$; OTB100 and UAV123 report AUC. Lower values indicate stronger attacks.

Dataset	GOT10k			OTB	UAV
Metric / Deployment order	AO	SR0.5	SR0.75	AUC	AUC
$0 \to 1 \to 2 \to 3 \to 4$	75.9	88.7	67.5	63.1	62.4
$4 \to 3 \to 2 \to 1 \to 0$	78.1	91.2	71.6	64.4	63.1
$2 \to 0 \to 4 \to 1 \to 3$	78.2	91.2	71.9	64.5	62.1
$1 \to 3 \to 0 \to 2 \to 4$	77.8	90.8	71.2	64.3	62.7

I More Ablation

As illustrated in Fig. 10, sequence-level analysis further interprets the internal working mechanism of each module. The AO curves of the baseline and the variant without GIoU constraint reveal that the latter suffers drastic AO fluctuations with frequent local peaks, while the baseline curve declines steadily. This phenomenon proves that GIoU constraint anchors adversarial patches on valid target regions and prevents oscillating attack performance caused by meaningless disturbance. The histogram comparing the baseline and the setup excluding switching gain shows remarkable metric increments after removing this component, among which SR@0.5 gains the most improvement, highlighting the indispensable role of switching gain for stable long-sequence adversarial attack. Moreover, the bar chart of AO degradation between baseline and static single-patch attack intuitively reflects the performance gain from dynamic patch design, where over 5% AO drop is observed on multiple test sequences and validates that temporal evolution continuously strengthens attack robustness across diverse scenarios.

Overall, ablation results verify the necessity and complementarity of all building blocks in our dynamic adversarial patch framework. Specifically, dynamic patch establishes the temporal evolution foundation, switching gain guarantees uninterrupted adversarial interference, temporal memory accumulates inherited adversarial priors, the feature adapter refines progressive perturbation transfer, and GIoU constraint enhances the localization precision of perturbations. Benefiting from the coordinated effect of all modules, our method achieves persistent and high-efficiency adversarial attacks against history-prior enhanced trackers.

J Texture Evolution of the Learned Dynamic Patches

Digital-Domain Patches The five patches learned for each tracker exhibit an ordered rather than independent texture evolution, as shown in Fig. 11. Later patches repeatedly preserve dominant color regions, contour orientations, and local structures that already appear in earlier patches, while introducing changes in their boundaries, contrast, and fine-scale details. This visual continuity is consistent with adversarial-prior inheritance: the generation of p_k starts from the preceding feature and the accumulated historical state, so an effective adversarial pattern can be carried forward instead of being reconstructed independently at every step. The amount of visible change is tracker dependent. For some trackers, successive patches retain a common large-scale structure but modify local high-frequency components; for others, the inherited structure already produces a stable attack and only

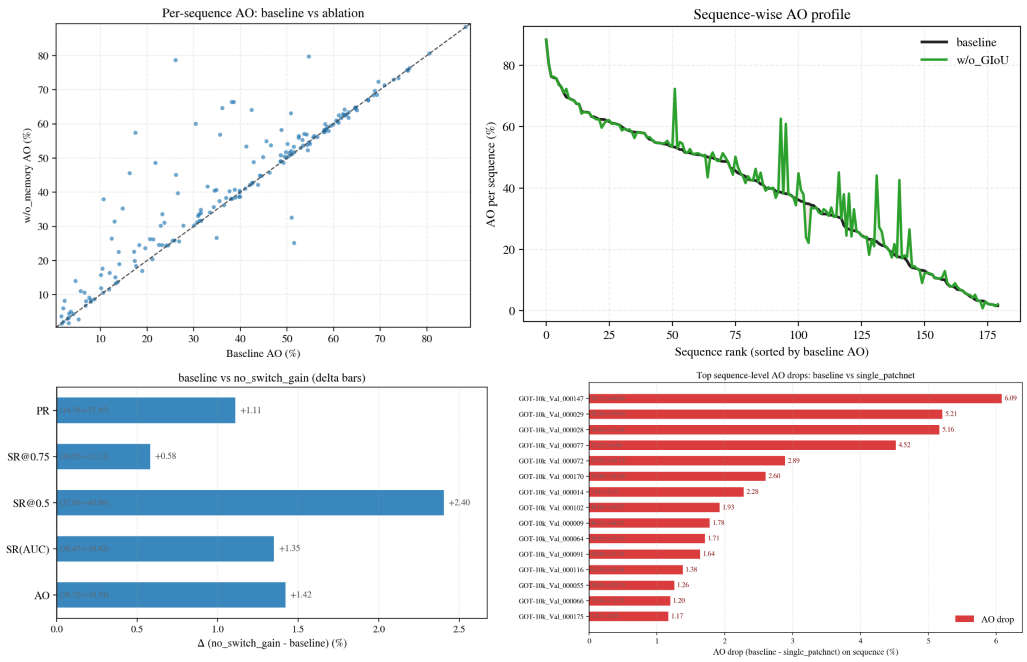


Figure 10: Sequence-level analysis of each core module. The compact single-column visualization compares per-sequence AO behavior, GIoU-constraint stability, switching-gain contribution, and dynamic-patch improvement.

limited refinement is required. The patch sequence therefore represents a progressive adversarial trajectory rather than a collection of unrelated static textures.

Physical-Domain Patches The physically optimized patches preserve the same inheritance pattern after spatio-temporal degradation augmentation. Recurring shapes and color transitions remain visible across the ordered sequence, indicating that later patches retain structures that survive frame-rate variation, trailing, glare, blur, and resolution degradation. Compared with purely digital optimization, several physical-domain sequences concentrate more strongly on broad, high-contrast regions and persistent edges, although the exact texture remains determined by the victim tracker. These structures are less dependent on individual high-frequency pixels and can remain recognizable after display and camera capture.

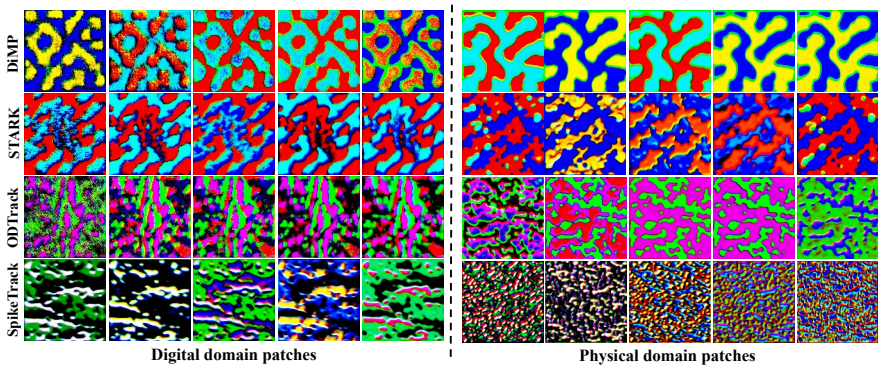


Figure 11: Display of digital domain and physical domain dynamic adversarial patches by DyPatch for different trackers.

Some adjacent patches appear nearly identical, particularly when a tracker can already be disrupted by a small family of closely related patterns. This is an expected outcome of the optimization objective

rather than a failure of the generator. DyPatch does not impose an auxiliary diversity loss that forces consecutive patches to look different; it optimizes tracking degradation and the effectiveness of each transition. Once an inherited texture provides a sufficiently strong adversarial direction, replacing it with a visually unrelated pattern offers little benefit, so the generator retains the dominant structure and introduces only the local variation needed by the subsequent patch. The resulting similarity suggests that the effective adversarial space of a particular tracker can be narrower than the full visual space of possible patches. At the same time, the patches are not exact copies: changes in local boundaries, colors, and fine structures preserve transition-specific interference while maintaining the adversarial prior accumulated earlier in the sequence.