Physical Adversarial Textures That Fool Visual Object Tracking

Rey Reza Wiyatno Anqi Xu
Element AI
Montreal, Canada
{rey.reza, ax}@elementai.com

Abstract

We present a method for creating inconspicuous-looking textures that, when displayed as posters in the physical world, cause visual object tracking systems to become confused. As a target being visually tracked moves in front of such a poster, its adversarial texture makes the tracker lock onto it, thus allowing the target to evade. This adversarial attack evaluates several optimization strategies for fooling seldom-targeted regression models: non-targeted, targeted, and a newly-coined family of guided adversarial losses. Also, while we use the Expectation Over Transformation (EOT) algorithm to generate physical adversaries that fool tracking models when imaged under diverse conditions, we compare the impacts of different scene variables to find practical attack setups with high resulting adversarial strength and convergence speed. We further showcase that textures optimized using simulated scenes can confuse real-world tracking systems for cameras and robots.

1. Introduction

Research on adversarial attacks [24, 9, 18] have shown that deep learning models, e.g., for classification and detection tasks, are confused by adversarial examples: slightly-perturbed images of objects that cause them to make wrong predictions. While early attacks digitally modified inputs to a victim model, later advances created photos [14] and objects in the physical world that lead to misclassification under diverse imaging conditions [7, 1]. Due to these added complexities, many physical adversaries were not created to look *indistinguishable* from regular items, but rather as *inconspicuous* objects such as colorful eyeglasses [20, 21].

We study the creation of physical adversaries for an object tracking task, of which the goal is to find the bounding-box location of a target in the current camera frame given its location in the previous frame. We present a method for generating Physical Adversarial Textures (PAT) that, when displayed as advertisement or art posters, cause regression-based neural tracking models like GOTURN [10] to break away from their tracked targets, even though these textures do not look like targets to human eyes, as seen in Figure 1.

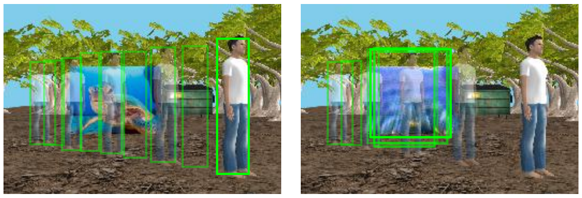

(a) source texture (b) adversarial texture

Figure 1: A poster of a Physical Adversarial Texture resembling a photograph, causes a tracker's bounding-box predictions to lose track as the target person moves over it.

Fooling a tracking system comes with added challenges compared to attacking classification or detection models. Since a tracker adapts to changes in the target's appearance, an adversary must be universally effective as the target moves and turns. Also, some trackers like GOTURN only search within a sub-region of the frame around the previous target location, and so only a small part of the PAT may be in view and not obstructed, yet it must still be potent. Furthermore, it is insufficient for the tracker to be slightly off-target on any single frame, as it may still end up tracking the target semi-faithfully; robust adversaries must cause the system to break away from the tracked target over time.

Our main contributions are as follows:

1. first known demo of adversaries for sequential tracking tasks, impacting domains such as surveillance, drone photography, and autonomous convoying,
2. coining of "guided adversarial losses" concept, which strikes a middle-ground between targeted and non-targeted adversarial objectives, and empirically shown to enhance convergence and adversarial strength,
3. study of Expectation Over Transformation (EOT) [1], highlighting the need to randomize only certain scene variables while still creating potent adversaries, and
4. show sim-to-real transfer of PATs created using a non-photorealistic simulator and diffuse-only materials.

2. Related Work

Early *white-box* physical adversarial attacks, which assumed access to the victim model's internals, created printable adversaries that were effective under somewhat varying views [14], by using gradient-based methods such as FGSM [9]. Similar approaches were employed to create eyeglass frames for fooling face recognition models [20, 21], and to make stop signs look like speed limits to a road sign classifier [7]. Both latter systems only updated gradients within a *mask*ed region in the image, namely over the eyeglass frame or road sign. Still, neither work explicitly accounted for the effects of lighting on the imaged items.

Expectation Over Transformation (EOT) [1] formalized the strategy used by [20, 7] of optimizing for adversarial attributes of a mask, by applying a combination of random transformations to it. By varying the appearance and position of a 2-D photograph or 3-D textured object as the mask, EOT-based attacks [1, 3, 15] generated physically-realizable adversaries that are robust within a range of viewing conditions. Our attack also applies EOT, but we importantly study the efficacy and the need to randomize over different transformation variables, including foreground/background appearances, lighting, spatial locations of the camera, target, adversary, and surrounding objects.

CAMOU is a *black-box* attack that also applied EOT to create adversarial textures for a car that made it non-detectable by object detection networks. CAMOU approximated the gradient of an adversarial objective through both the complex rendering process and opaque victim network, by using a learned surrogate mapping [17] from the texture space directly onto the detector's confidence score. Both their attack and evaluations were carried out using a photo-realistic rendering engine. Still, this method was not tested in the real world, and also incurs high computational costs and potential instability risks due to the alternation optimizing the surrogate model and the adversarial perturbations.

DeepBillboard [27] attacked autonomous driving systems by creating adversarial billboards that caused a victim model to deviate its predicted steering angles within real-world drive-by sequences. While our work shares many commonalities with DeepBillboard, we confront added challenges by attacking a sequential tracking model rather than a per-frame regression network, and we also contrast the effectiveness of differing adversarial objectives.

3. Object Tracking Networks

Various learning-based tracking methods have been proposed, such as the recent GOTURN [10] deep neural network that regresses the location of an object in a camera frame given its previous location and appearance. While other tracking methods based on feature-space cross-correlation [2, 25] and tracking-by-detection [8] are also viable, we focus on GOTURN models to ground our studies on the effectiveness of different types of adversarial losses, as well as the compute efficiency of an EOT-based attack.

As seen in Figure 2, given a target's bounding-box location $\hat{l}_{j-1}$ of size $w \times h$ in the previous frame f_{j-1}, GOTURN crops out the *template* $\tilde{f}_{j-1}$ as a region of size $2w \times 2h$ around the target within f_{j-1}. The current frame f_j is also cropped to the same region, yielding the *search area* $\tilde{f}_j$, which is assumed to contain most of the target still. Both the template and search area are resized to 227×227 and processed through convolutional layers. The resulting feature maps are then concatenated and passed through fully-connected layers with non-linear activations, ultimately regressing $l_j = \{(x_{min}, y_{min}), (x_{max}, y_{max})\} \in [0, 1]^4$, that is, the top-left and bottom-right coordinates of the target's location within the current search area $\tilde{f}_j$.

Such predictions can also be used for visual servoing, i.e., to control an aerial or wheeled robot to follow a target through space. One approach [11, 22] is to regulate the center-points and areas of predictions about the center of the camera frame and the desired target size, respectively, using Proportional-Integral-Derivative (PID) controllers on the forward/backward, lateral, and possibly vertical velocities of the vehicle. In this work, we show that visual tracking models, as well as derived visual servoing controllers for aerial robots, can be compromised by PATs.

4. Attacking Regression Networks

For classification tasks, an adversarial example is defined as a slightly-perturbed version of a source image that satisfies two conditions: *adversarial output* — the victim model misclassifies the correct label, and *perceptual similarity* — the adversary is perceived by humans as similar to the source image. We discuss necessary adjustments to both conditions when attacking regression tasks. While recent work has shown the existence of adversaries that confuse regression tasks [6, 27], there is still a general lack of analysis on the strength and properties of adversaries as a function of different attack objectives. In this work, we consider various ways to optimize for an adversary, and notably formalize a new family of *guided* adversarial losses. While this work focuses on images, the concepts discussed below are generally applicable to other domains as well, such as fooling audio transcriptions [6].

4.1. Adversarial Strength

There is no task-agnostic analog to misclassification for regression models, due to the non-discrete representation of their outputs. Typically, a regression output is characterized as adversarial by thresholding a task-specific error metric. This metric may also be used to quantify *adversarial strength.* For instance, adversaries for human pose-prediction can be quantified by the percentage of predicted joint poses beyond a certain distance from ground-truth locations [6]. As another example, DeepBillboard [27] defines unsafe driving for an autonomous vehicle as experiencing an excessive amount of total lateral deviation, and quantifies adversarial strength as the percentage of frames in a given unit of time where the steering angle error exceeds a corresponding threshold.

When fooling a visual tracker, the end-goal is for the system to break away from the target *over time*. Therefore, we consider a sequence of frames $F^\dagger = \{f_1^\dagger, f_2^\dagger, ..., f_N^\dagger\}$ where the target moves across a poster containing an adversarial texture χ, and quantify adversarial strength by the average amount of overlap between tracker predictions l_j (computed from $f_{j-1}^\dagger, f_j^\dagger$) and the target's actual locations $\hat{l_j}$. We also separate the tracker's baseline performance from the effects of the adversary, by computing the average overlap ratio across another sequence $F = \{f_1, f_2, ..., f_N\}$, in which the adversarial texture is replaced by an *inert source texture*. Thus, in this work, adversarial strength is defined by averaging the **mean-Intersection-Over-Union-difference** metric, $\mu IOUd$, over multiple generated sequences:

$$\begin{aligned} IOU(l_j, \hat{l_j}) =& \frac{\mathcal{A}(l_j \cap \hat{l_j})}{\mathcal{A}(l_j) + \mathcal{A}(\hat{l_j}) - \mathcal{A}(l_j \cap \hat{l_j})} \\ \mu IOUd =& \frac{1}{N-1} \sum_{j \in [2,N], f_j \in F} IOU\left(l_j(f_{j-1}, f_j), \hat{l_j}\right) \quad (1) \\ & - \frac{1}{N-1} \sum_{j \in [2,N], f_j^\dagger \in F^\dagger} IOU\left(l_j(f_{j-1}^\dagger, f_j^\dagger), \hat{l_j}\right) \end{aligned}$$

where $\cap$ denotes the intersection of two bounding boxes and $\mathcal{A}(\cdot)$ denotes the area of the bounding box l.

4.2. Perceptual Similarity

Perceptual similarity is often measured by the distance between a source image and its perturbed variant, e.g., using Euclidean norm in the RGB colorspace [24, 4]. Sometimes, we apply a loose threshold to this constraint, to generate universal adversaries that remain potent under diverse conditions [16, 1, 26]. Other times, the goal is not to *imitate* a source image, but merely to create an *inconspicuous* texture that does not look harmful to humans, yet cause models to misbehave [20, 3, 27]. With this work, we aim to raise public awareness that *colorful-looking art can be harmful to vision models*.

4.3. Optimizing for Adversarial Behaviors

While our attack's end-goal is to cause the tracker to break away from its target, we can encourage different *adversarial behaviors*, such as locking onto part of an adversarial poster or focusing onto other parts of the scene. These behaviors are commonly optimized into an adversary through loss minimization, e.g., using gradient descent. The literature has proposed several families of adversarial losses, notably:

- the baseline **non-targeted** loss $\mathcal{L}_{nt}$ maximizes the victim model's training loss, thus causing it to become generally confused (e.g., FGSM [9], BIM [14]);
- **targeted** losses $\mathcal{L}_t$ also apply the victim model's training loss, but to minimize the distance to an *adversarial target output* (e.g., JSMA [18]);
- we define **guided** losses $\mathcal{L}_g$ as middle-grounds between $\mathcal{L}_{nt}$ and $\mathcal{L}_t$, which regulate specific adversarial *attributes* rather than strict output values, analogous to misclassification onto a set of output values [14]; and
- **hybrid** losses use a weighted linear combination of the above losses to gain adversarial strength and speed up the attack (e.g., C&W [4], Hot/Cold [19] attacks).

The motivation for guided losses stems from our observations of the optimization rigidity of targeted losses, and weak guidance from the non-targeted loss. Although similar ideas have been used [4, 27], we formally coin "guided adversarial objectives" as those that regulate attributes of the victim model's output about specific adversarial values.

To fool object trackers, we consider these specific losses:

- $\mathcal{L}_{nt} = -||l_j^\dagger - \hat{l_j}||_1$ increases GOTURN's training loss;
- $\mathcal{L}_{t-} = ||l_j^\dagger - \{(0.0, 0.9), (0.1, 1.0)\}||_1$ shrinks predictions towards the bottom-left corner of the search area;
- $\mathcal{L}_{t=} = ||l_j^\dagger - \{(0.25, 0.25), (0.75, 0.75)\}||_1$ predicts the exact location of the target in the previous frame;
- $\mathcal{L}_{t+} = ||l_j^\dagger - \{(0.0, 0.0), (1.0, 1.0)\}||_1$ grows predictions to the maximum size of the search area;
- $\mathcal{L}_{ga-} = \min(\mathcal{A}(l_j^\dagger) - \mathcal{A}(\hat{l_j}), 0)$ encourages the area of each prediction to shrink from the ground-truth value;
- $\mathcal{L}_{ga+} = \max(\mathcal{A}(l_j^\dagger) - \mathcal{A}(\hat{l_j}), 0)$: encourages the area of each prediction to grow from the ground-truth value.

Note that other guided losses are also possible, such as maximizing or minimizing the magnitudes of predictions. For succinctness, we evaluated against a non-targeted loss and the simplest of targeted losses as baselines, to show that a well-engineered guided loss has the potential for better convergence and adversarial strength.

Additionally, we can enforce perceptual similarity by adding a Lagrangian-relaxed loss $\mathcal{L}_{ps}$ [24, 4, 1]. Its associated weight can be set heuristically, or fine-tuned via line search into the smallest value resulting in sufficient adversarial strength. While most of our experiments generate *inconspicuous* adversaries that do not enforce perceptual similarity, Section 6.4 specifically showcases *imitation* attacks.

In summary, our attack method optimizes a (possibly-imitated) source texture χ_0 into an adversarial variant χ_i over $i \in [1, I_{max}]$ iterations, by minimizing a weighted linear combination of loss terms:

$$\mathcal{L} = \bar{w} \cdot [\mathcal{L}_{nt}, \mathcal{L}_{t...}, \mathcal{L}_{g...}, \mathcal{L}_{ps}]^T \quad (2)$$

where the texture is incrementally updated as:

$$\chi_i = \chi_{i-1} + \alpha_i \cdot \Delta\chi \quad (3)$$

Here, α_i denotes the step size at the i-th iteration, and $\Delta\chi$ denotes a perturbation term based on the gradient $\nabla_\chi \mathcal{L}$.

5. Physical Adversarial Textures

We now discuss how the above attack formulation can be generalized to produce Physical Adversarial Textures (PAT) that resemble colorful art. Such PATs, when displayed on a digital poster and captured by camera frames near a tracked target, causes a victim model to lose track of the target.

In this work, we assume to have *white-box* access to the GOTURN network's weights and thus the ability to back-propagate through it. We focus on tracking people and humanoid robots in particular and assume that the tracker was trained on such types of targets.

As mentioned in Section 1, several challenges arise when creating adversaries to fool temporal tracking models. We address these by applying the Expectation Over Transformation (EOT) algorithm [1], which minimizes the expected loss $\mathbb{E}[\mathcal{L}]$ over a minibatch of B scenes imaged under diverse conditions. EOT marginalizes across the distributions of different transformation variables, such as the poses of the camera, tracked target, and poster, as well as the appearances of the target, environmental surroundings, and ambient lighting. However, marginalizing over wide ranges of condition variables can be very computationally expensive. Thus, Section 6.3 studies the effects on adversarial strength and attack speeds resulting from varying EOT variables.

An essential addition when generating a physical adversarial item, as opposed to a digital one, is the need to render the textured item into scenes as it evolves during the attack process. Our attack creates PATs purely from scenes rendered using the Gazebo simulator [13], yet Section 6.5 will show that these adversaries are also potent in the real world.

5.1. Modeling rendering and lighting

To optimize the loss with respect to the texture of a physical poster, we need to differentiate through the rendering process. Rendering can be simplified into two steps: *projecting* the texture onto the surface of a physical item and then onto the camera's frame, and *shading* the color of each frame pixel depending on light sources and material types.

Similar to [15], we sidestep shading complexities, such as spotlight gradients and specular surfaces, by assuming controlled imaging conditions: the PAT is displayed on a matte material and is lit by a far-away sun-like source, and the camera's exposure is adjusted not to cause pixel saturation. Consequently, we employ a linear lighting model, where each pixel's RGB intensities in the camera frame is a scaled and shifted version of pixel values for the projected texture coordinate. During our attack, we query the Gazebo simulation software to obtain exact gains for light intensity and material reflectance, while before each real-world test we fit parameters of this per-channel linear lighting model once, using a displayed color calibration target.

As for the projection component, we modified Gazebo's renderer to provide projected frame coordinates for each texture pixel (similar to [1]), as well as occlusion masks and bounding boxes of the target in the foreground. We then use this texture-to-frame mapping to manually back-propagate through the projection process onto the texture space.

5.2. PAT Attack

Figure 2 shows the overall procedure for generating a Physical Adversarial Texture. Starting from a source texture χ_0, we perform minibatch gradient descent on $\mathcal{L}$ to optimize pixel perturbations that adds onto the texture, for a total of I_{max} iterations. On each iteration i, we apply EOT to a minibatch of B scenes, each with randomized settings for the poses of the camera, target, and poster, the identities of the target and background, and the hue-saturation-value settings of a single directional light source.

Each scene entails two frames $\{f_{j-1}, f_j\}$, in which both the camera and tracked target may have moved between the previous and current frames. Given the target's previous *actual* location $\hat{l}_{j-1}$, we crop both frames around a correspondingly scaled region, then resize and process them through the GOTURN network, to predict the bounding-box location l_j of the target in the current frame. We then back-propagate from the combined loss objective $\mathcal{L}$ onto the texture space through all partial-derivative paths. After repeating the above process for all B scenes, we compute the expected texture gradient, and update the texture using the Fast Gradient Sign optimizer [9], scaled by the current iteration's step size α_i:

$$\Delta\chi = -sign(\nabla_\chi \mathbb{E}[\mathcal{L}]) \quad (4)$$

6. Experiments

In this section, we present an empirical comparison of PAT attacks using non-targeted, targeted, guided, and hybrid losses. We also assess which EOT conditioning variables are most useful for producing strong adversaries quickly. Furthermore, we analyze PATs resulting from imitation attacks and their induced adversarial behaviors. Finally, we showcase the transfer of PATs generated in simulation for fooling tracking system in a real-world setup.

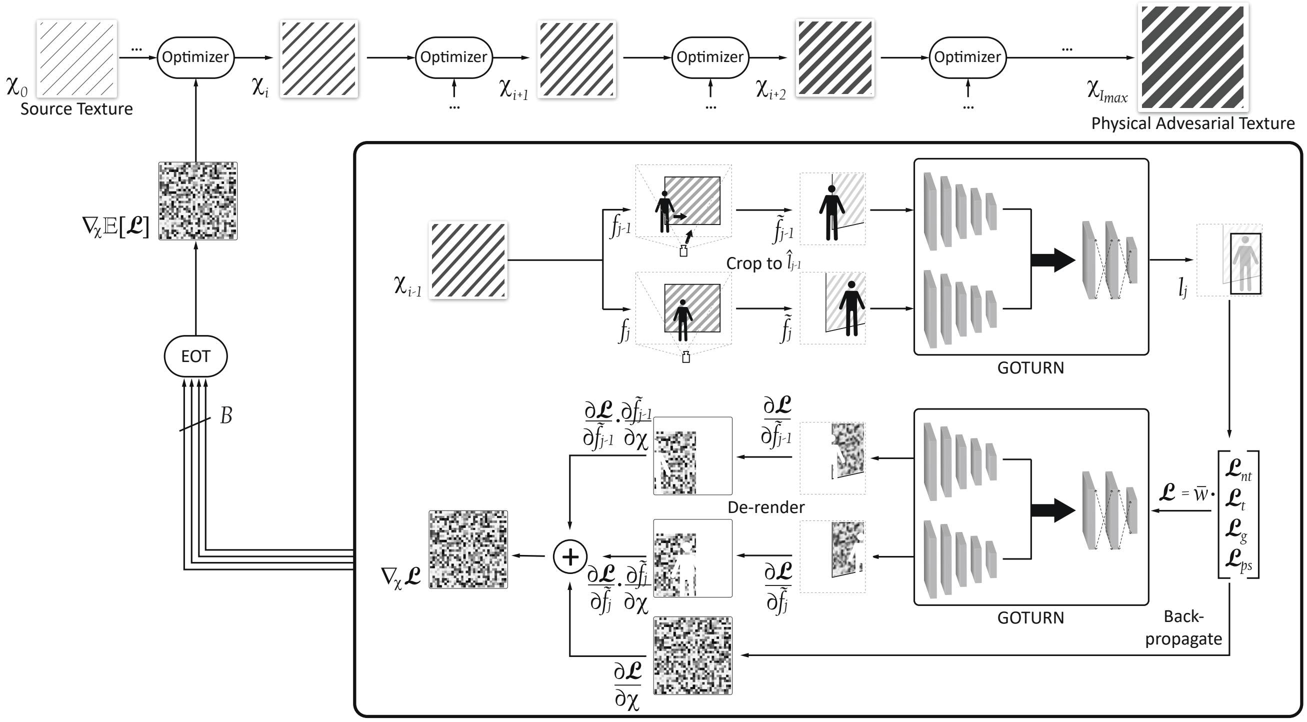

χ_0
Source Texture
Optimizer
χ_i
χ_{i+1}
χ_{i+2}
$\chi_{I_{max}}$
Physical Advesarial Texture
$\nabla_\chi \mathbb{E}[\mathcal{L}]$
EOT
B
χ_{i-1}
f_{j-1}
$\tilde{f}_{j-1}$
Crop to $\hat{l}_{j-1}$
f_j
$\tilde{f}_j$
GOTURN
l_j
$\frac{\partial \mathcal{L}}{\partial \tilde{f}_{j-1}} \cdot \frac{\partial \tilde{f}_{j-1}}{\partial \chi}$
$\frac{\partial \mathcal{L}}{\partial \tilde{f}_{j-1}}$
De-render
$\mathcal{L} = \bar{w} \cdot \begin{bmatrix} \mathcal{L}_{nt} \\ \mathcal{L}_t \\ \mathcal{L}_g \\ \mathcal{L}_{ps} \end{bmatrix}$
$\nabla_\chi \mathcal{L}$
$\frac{\partial \mathcal{L}}{\partial \tilde{f}_j} \cdot \frac{\partial \tilde{f}_j}{\partial \chi}$
$\frac{\partial \mathcal{L}}{\partial \tilde{f}_j}$
GOTURN
Back-propagate
$\frac{\partial \mathcal{L}}{\partial \chi}$

Figure 2: The Physical Adversarial Texture (PAT) Attack creates adversaries to fool the GOTURN tracker, via minibatch gradient descent to optimize various losses, using randomized scenes following Expectation Over Transformation (EOT).

6.1. Setup

All PAT attacks were carried out using simulated scenes rendered by Gazebo. This conveniently provides an endless stream of independently-sampled scenes, with controlled poses and appearances for the target, textured poster, camera, background, and lighting. We created multiple scenarios, including 3 outdoor views of a 2.6m × 2m poster in front of a building, forest, or playground, and an indoor coffee shop scene where a half-sized poster is hung on the wall. We also varied tracked targets among models of 3 different persons and 2 humanoid robots.

6.1.1 Trained GOTURN models

We trained several GOTURN networks on various combinations of synthetic and real-world labeled datasets for tracking people and humanoid robots. The synthetic dataset contains over $1,400$ short tracking sequences with more than $300,000$ total frames, while the real-world dataset consists of 29 videos with over $50,000$ frames of one of two persons, moving around an office garage and at a park. We used the Adam optimizer [12] with an initial learning rate of 10^{-5} and a batch size of 32. Models trained on synthetic-only data (`sim`) lasted $300,000$ iterations with the learning rate halved every $30,000$ iterations, while those trained on combined datasets (`s+r`) or on the real-world dataset after bootstrapping from the synthetic-trained model (`s2r`) ran for $150,000$ iterations with the learning rate halved every $15,000$ iterations. In addition to the architecture of [10] (`Lg`), we also trained smaller-capacity models with more aggressive striding instead of pooling layers and fewer units in the fully-connected layers (`Sm`). While this section evaluates a subset of model instances, our Appendices present comprehensive results on other networks.

6.1.2 Evaluation Metric

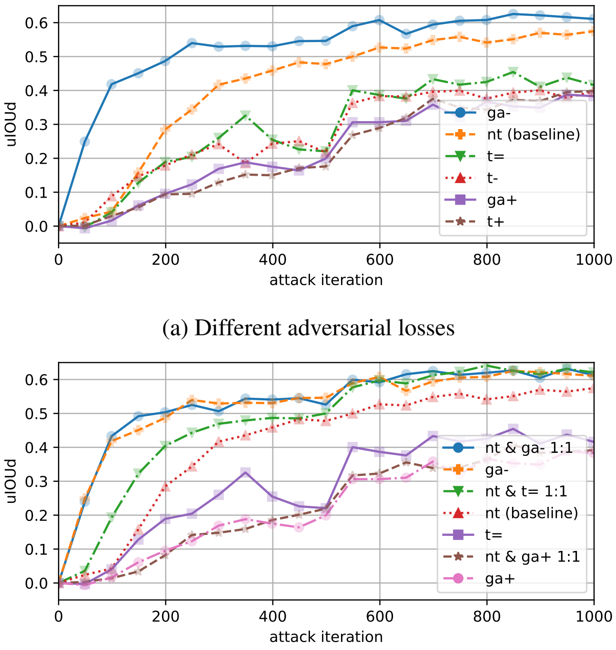
As discussed in Section 4.1, we evaluate each PAT by generating sequences in which a tracked target moves from one side of the textured poster to the other. Each sequence randomly draws from manually-chosen ranges for the target, camera, and poster poses, hue-saturation-value settings for the light source, target identities, and background scenes. We run the GOTURN tracker on each sequence twice, differed by the display of either the PAT or an inert source texture on the poster. Adversarial strength is then computed as the average $\mu IOUd$ metric over 20 random sequence pairs.

Anecdotally, for average $\mu IOUd$ values around 0.2, the tracker's predictions expanded and worsened as the target moved over the poster, yet GOTURN locked back onto the target as it moved away. In contrast, values greater than 0.4 reflected cases where GOTURN consistently lost track of the target during and at the end of the sequence, thus showing notably worse tracking compared to an inert poster.

6.1.3 Baseline Attack Settings

We carried out hyperparameter search to determine a set of attack parameters that produce strong adversaries (see Appendix D). Unless otherwise stated, each PAT attack ran on the regular-capacity synthetic-trained GOTURN model (`Lg,sim`), with: $I_{max} = 1,000$ attack iterations, EOT minibatch with $B = 20$ samples, FGS optimizer with step sizes of $\alpha_{i \leq 500} = 0.75$ and then $\alpha_{i > 500} = 0.25$, and starting from a randomly-initialized source texture with 128×128 pixels. All presented results are averaged over 10 attack instances, with different initial random seeds.

6.2. Efficacy of Adversarial Losses for Regression



(a) Different adversarial losses

(b) Individual vs hybrid adversarial losses

Figure 3: PAT attack strength for various adversarial losses.

Figure 3a depicts the progression in adversarial strength throughout PAT attack runs for the different adversarial losses proposed in Section 4.3. Comparing against the non-targeted baseline EOT attack ($\mathcal{L}_{nt}$) , most targeted and guided losses resulted in slower convergence and worse final adversarial strength. This is not surprising as these adversarial objectives apply stricter constraints on the desired adversarial behaviors and thus need to be optimized for longer. As the sole exception, the guided loss encouraging smaller-area predictions ($\mathcal{L}_{ga-}$) attained the fastest convergence and best adversarial strength overall. This suggests that *well-engineered adversarial objectives, especially loosely-guided ones, benefit by speeding up and improving the attack process* on regression tasks.

In Figure 3b, we see that combining $\mathcal{L}_{nt}$ with most targeted or guided losses did not significantly change performance. While not shown, we saw similar results when using 1:1000 weight ratios. However, the 1:1 combination of $\mathcal{L}_{nt}$ & $\mathcal{L}_{t=}$ attained better overall performance than both $\mathcal{L}_{nt}$ and $\mathcal{L}_{t=}$. This suggests that *sometimes adding a non-targeted loss to a targeted or guided one helps*, possibly due to the widening of conditions for adversarial behaviors.

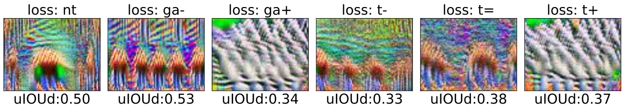


Figure 4: PATs generated using different adversarial losses.

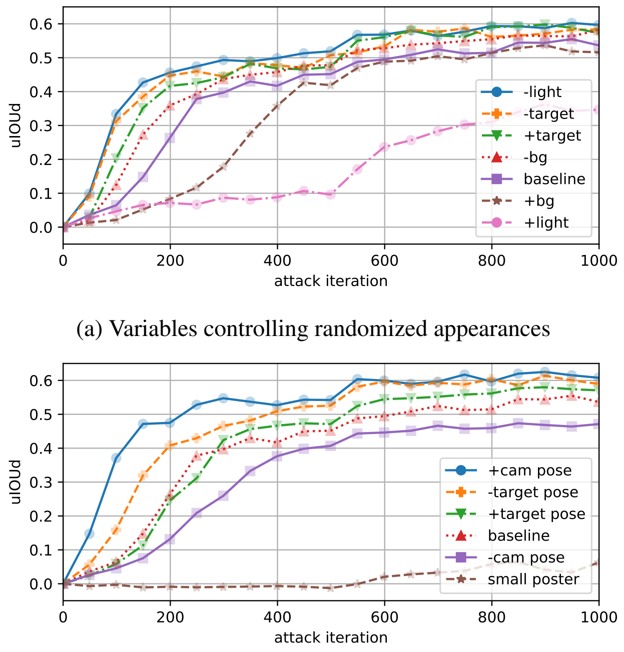
As seen in Figure 4, various patterns emerge in PATs generated by different losses. We note that dark "striped patches" always appeared in PATs generated from certain losses, and these patches caused GOTURN to lock on and break away from the tracked target. On the other hand, "striped patches" did not show up for PATs created using $\mathcal{L}_{ga+}$ or $\mathcal{L}_{t+}$, which showed uniform patterns. This is expected as these losses encourage the tracker's predictions to grow in size, rather than fixating onto a specific location.

6.3. Ablation of EOT Conditioning Variables

Here, we assess which variables for controlling the random sampling of scenes had strong effects, and which ones could be set to fixed values without impact, thus reducing scene randomization and speeding up EOT-based attacks.

As seen in Figure 5a, reducing variety in appearances of the background (`-bg`), target (`-target`), and light variations (`-light`), did not substantially affect adversarial strength when other parameter ranges were held constant. Also, increasing diversity in `+target` and `+bg` did not result in different end-performance. This suggests that *diversity in target and background appearances do not strongly affect EOT-based attacks*. On the other hand, `+light` converged much slower than other settings. Thus, we conclude that *if randomized lighting is needed to generalize the robustness of PATs during deployment, then more attack iterations are needed to ensure convergence*.

For pose-related variables in Figure 5b, halving the poster size (`small poster`) caused the PAT attack to fail. Changing the ranges of camera poses (`+cam pose`, `-cam pose`) resulted in notable performance differences, therefore we note that *more iterations are needed to generate effective PATs under wider viewpoint ranges*. Perhaps surprisingly, for `-target pose`, *locking the target's pose to the center of the poster resulted in faster and stronger convergence*. This is likely because regions around the static target obtained consistent perturbations across all scenes, and so developed adversarial patterns faster.



(a) Variables controlling randomized appearances

(b) Variables controlling randomized poses

Figure 5: PAT attack strength for various EOT variables.

6.4. Imitation Attacks

As discussed in Section 4.3, we can add a perceptual similarity loss term to make the PAT imitate a meaningful source image. A larger perceptual similarity weight w_{ps} perturbs the source less, but at the cost of slower convergence and weaker or ineffective adversarial strength. Results below reflect a manually-tuned setting of $w_{ps} = 0.6$.

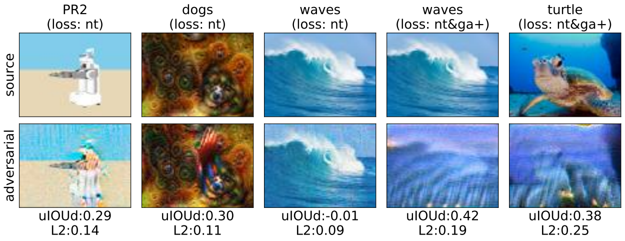
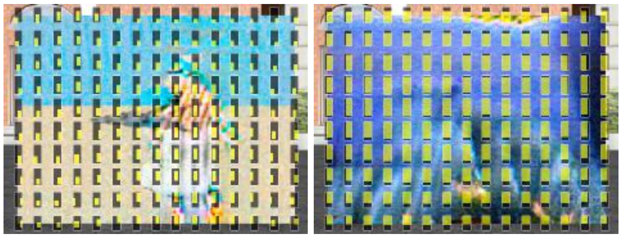


Figure 6: Adversarial imitations under various losses.

Figure 6 shows that some source images, coupled with the right adversarial loss, led to stronger imitations than others. For instance, the waves source was optimized into a potent PAT using $\mathcal{L}_{nt}$ & $\mathcal{L}_{ga+}$, yet using $\mathcal{L}_{nt}$ alone failed to produce an adversarial texture. However, we found that for a given threshold on L_2 distance, guided losses generally converged faster to reach potent behaviors, yet suffered from weakened adversarial strength compared to $\mathcal{L}_{nt}$ over prolonged attack iterations (see Appendix F for quantitative details). Also, under larger w_{ps} constraints, we saw that adversarial perturbations appeared only in selective parts of the texture. Notably, the "striped patches" seen in non-imitated PATs (Figure 4) also emerged near the dogs' face and over the PR2 robot, when optimized using $\mathcal{L}_{nt}$. We thus conclude that the PAT attack produces *critical adversarial patterns* such as these patches first, and then perturbs other regions into *supporting adversarial patterns*.

Further substantiating this claim, Figure 7 visualizes predicted bounding-boxes within search areas located at different sub-regions of PATs. We see from Figure 7a that predictions around the adversarial "striped patch" made GOTURN track towards it. This suggests that such *critical adversarial patterns induce potent lock-on behaviors that break tracking, regardless of where the actual target is positioned.* On the other hand, shown in Figure 7b, the "regular wavy" pattern optimized using $\mathcal{L}_{ga+}$ resulted in the intended adversarial behavior of larger-sized predictions, regardless of the search area's location.



(a) Lg, sim tracker; $\mathcal{L}_{nt}$ loss (b) Lg, s+r tracker; $\mathcal{L}_{ga+}$ loss

Figure 7: Adversarial behaviors emerging from PATs.

6.5. Demonstration of Sim-to-real Transfer

To assess the real-world effectiveness of PATs generated purely using simulated scenes, we displayed them on a 50″ TV within an indoor environment with static lighting. We carried out two sets of person-following experiments using the camera on a Parrot Bebop 2 drone: *tracking* sessions with a stationary drone, and *servoing* runs where the tracked predictions were used to control the robot to follow the target through space (see Section 3 for details).

In both experiments, we tasked the s+r GOTURN instance to follow people that were not seen in the tracker's training dataset. While we tested under different light intensities, for each static setting, we first fit a linear per-channel lighting model to a color calibration target, and then adjusted camera frames accordingly, as explained in Section 5.1. We carried out this *optional* step to showcase adversarial performance in best-case conditions, and note that none of the simulated evaluations corrected for per-scenario lighting. Also, this correction compensates for fabrication errors that may arise when displaying the PAT on a TV or printed as a static poster, and further serves as an alternative to adding a Non-Printability Score to the attack loss [20].

During our experiments, we observed $57/80$ stationary runs and $6/18$ servoing runs to have strong lock-on adversarial behaviors. For succinctness, we focus on qualitative analyses below; please refer to Appendix H for more extensive quantitative results and visual samples.

For stationary tracking runs, only adversaries containing “striped patches” consistently made GOTURN break away from the person. Other PATs optimized by, e.g., $\mathcal{L}_{ga+}$, caused the tracker to make worse predictions as the target moved in front of the poster, yet it ultimately locked back onto the person. While these results were partially due to our limited-size digital poster, a more general cause is likely because such losses induced weak adversarial behaviors: by encouraging growing predictions, GOTURN could still see and thus track the person within an enlarged search area.

Returning to the best-performing PATs containing “striped patches”, the tracker strongly preferred to lock onto these rather than the person. Moreover, even though the person could regain GOTURN’s focus by completely blocking the patch, as soon as he or she moved away, the tracker locked back onto the patch, as seen in Figure 8. Furthermore, these physical adversaries were robust to various viewing distances and angles, and even for settings outside the ranges used to randomize scenes during the PAT attack.

Our servoing tests showed that it was generally harder to make GOTURN completely break away from the target. Since the drone was moving to follow the target, even though the tracker’s predictions were momentarily disturbed or locked onto the PAT, often the robot’s momentum caused GOTURN to return its focus onto the person. We attribute the worsened PAT performance to motion blurring, light gradients, and specular reflections that were present due to the moving camera, all of which were assumed away by our PAT attack. Nevertheless, we believe that these advanced scene characteristics can be marginalized by the EOT algorithm, using a higher-fidelity rendering engine than our implementation.

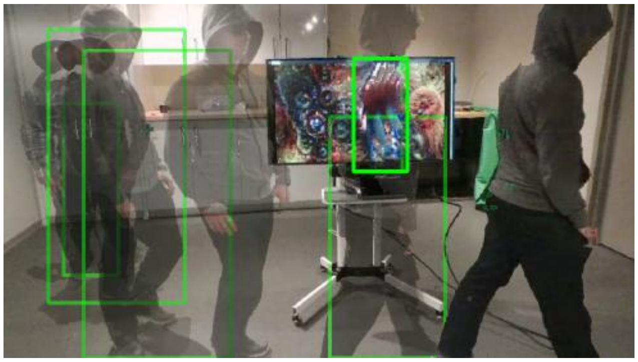


Figure 8: An imitated PAT, created in simulation, can fool a person-tracker in the real world.

Finally, we speculate that synthetically-generated adversarial patterns like the “striped patches” may look like simulated people or robot targets in GOTURN’s view. If so, then our real-world transfer experiments may have been aided by GOTURN’s inability to tell apart synthetic targets from real people. This caveat may be overcome by carrying out PAT attack using scenes synthesized with textured 3-D reconstructions or photograph appearances of the intended target.

7. Conclusion

We presented a system to generate Physical Adversarial Textures (PAT) for fooling object trackers. These “PATterns” induced diverse adversarial behaviors, emerging from a common optimization framework with the end-goal of making the tracker break away from its intended target. We compared different adversarial objectives and showed that a new family of guided losses, when well-engineered, resulted in stellar adversarial strength and convergence speed. We also showed that a naive application of EOT by randomizing *all* aspects of scenes was not necessary. Finally, we showcased synthetically-generated PATs that can fool real-world trackers.

We hope to raise awareness that inconspicuously-colored items can mislead modern vision-based systems *by merely being present in their vicinity*. Despite recent advances, we argue that purely vision-based tracking systems are not robust to physical adversaries, and thus recommend commercial tracking and servoing systems to integrate auxiliary signals (e.g., GPS and IMU) for redundancy and safety.

Since a vital goal of this work is to show the existence of inconspicuous patterns that fool trackers, we made the simplifying assumption of white-box access. More practically, it might be possible to augment the PAT attack using diverse techniques [17, 5, 23] to fool black-box victim models. Another improvement could be to directly optimize non-differentiable metrics such as $\mu IOUd$ by, e.g., following the Houdini method [6]. Finally, although the textures shown in this work may appear inconspicuous prior to our demonstrations, they are nevertheless clearly visible and thus can be detected and protected against. As the research community aims to defend against physical adversaries, we should continue to be on the lookout for potent PATs that more closely imitate natural items in the physical world.

Acknowledgements

We want to thank Dmitri Carpov, Matt Craddock, and Ousmane Dia for helping on the codebase implementation, Nicolas Chapados and Pedro Pinheiro for valuable feedback on our manuscript, and Minh Dao for helping with visual illustrations. We would also like to thank Philippe Beaudoin, Jean-François Marcil, and Sharlene McKinnon for participating in our real-world experiments.

References

[1] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *proceedings of the 35th International Conference on Machine Learning (ICML)*, Sweden, 2018.

[2] Luca Bertinetto, Jack Valmadre, João F Henriques, Andrea Vedaldi, and Philip H. S. Torr. Fully-convolutional siamese networks for object tracking. In *the European Conference on Computer Vision (ECCV) Workshops*, 2016.

[3] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *CoRR*, abs/1712.09665, 2017.

[4] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *the IEEE Symposium on Security and Privacy (S&P)*, 2017.

[5] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *proceedings of the 10th ACM Workshop on Artificial Intelligence and Security (AISec)*. ACM, 2017.

[6] Moustapha Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. In *advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[7] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[8] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect. In *proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[9] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[10] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[11] Dries Hulens and Toon Goedemé. Autonomous flying cameraman with embedded person detection and tracking while applying cinematographic rules. In *proceedings of the 14th Conference on Computer and Robot Vision (CRV)*, 2017.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[13] Nathan P. Koenig and Andrew Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[14] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *the International Conference on Learning Representations (ICLR) Workshops*, 2016.

[15] Hsueh-Ti Derek Liu, Michael Tao, Chun-Liang Li, Derek Nowrouzezahrai, and Alec Jacobson. Beyond pixel norm-balls: Parametric adversaries using an analytically differentiable renderer. In *proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[17] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *proceedings of the ACM on Asia Conference on Computer and Communications Security (ASIA CCS)*, 2017.

[18] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *the IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016.

[19] Andras Rozsa, Ethan M. Rudd, and Terrance E. Boult. Adversarial diversity and hard positive generation. In *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2016.

[20] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016.

[21] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. A general framework for adversarial examples with objectives. *ACM Transactions on Privacy and Security (TOPS)*, 2019.

[22] Florian Shkurti, Wei-Di Chang, Peter Henderson, Md Jahidul Islam, Juan Camilo Gamboa Higuera, Jimmy Li, Travis Manderson, Anqi Xu, Gregory Dudek, and Junaed Sattar. Underwater multi-robot convoying using visual tracking by detection. In *proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[23] Jiawei Su, Danilo V. Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 2019.

[24] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[25] Jack Valmadre, Luca Bertinetto, Joao Henriques, Andrea Vedaldi, and Philip H. S. Torr. End-to-end representation learning for correlation filter based tracking. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

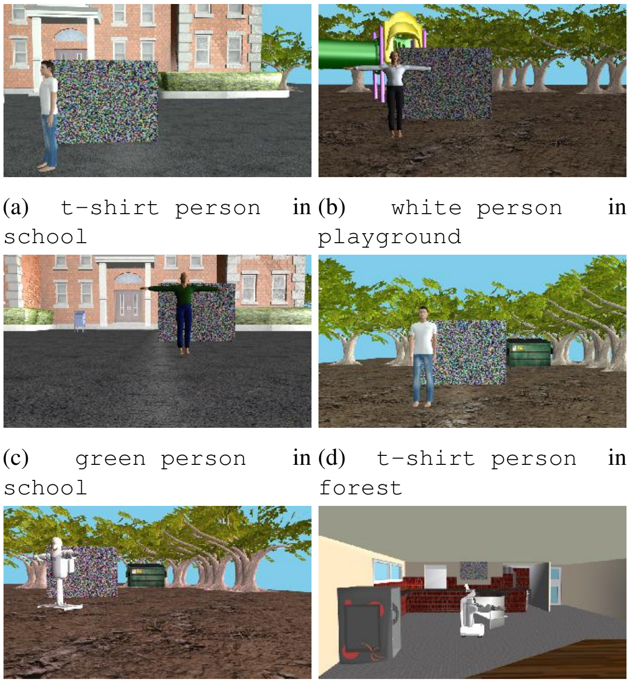
[26] Yang Zhang, Hassan Foroosh, Philip David, and Boqing Gong. CAMOU: Learning physical vehicle camouflages to adversarially attack detectors in the wild. In *proceedings of*

the International Conference on Learning Representations (ICLR), 2019.

[27] Husheng Zhou, Wei Li, Yuankun Zhu, Yuqun Zhang, Bei Yu, Lingming Zhang, and Cong Liu. Deepbillboard: Systematic physical-world testing of autonomous driving systems. *CoRR*, abs/1812.10812, 2018.

Appendix A. Simulated Scenarios

Figure 9 depicts samples of the 5 target (human or humanoid robot models) and 4 scenarios (outdoor and indoor scenes) that we created within the Gazebo simulation software [13]. These are used both for generating and evaluating Physical Adversarial Textures (PAT).



(a) `t-shirt person` in `school` (b) `white person` in `playground`

(c) `green person` in `school` (d) `t-shirt person` in `forest`

(e) `robonaut` in `forest` (f) `PR2` in `cafe`

Figure 9: Samples of simulated scenarios.

Appendix B. PAT Attack: Random Scene Configuration

The Expectation Over Transformation (EOT) algorithm [1] randomizes various parameters and aspects of scenes, such as camera placement and target appearance. By optimizing on these diverse and randomized scenes, we can ensure that the generated PAT would likely be universally adversarial. Table 1 presents *default* ranges used for continuous transformation variables used in our PAT Attack process, while Table 2 enumerates selections for discrete transformation variables. This default configuration is used in Sections 6.2, 6.4, and 6.5.

Table 1: Continous EOT variable ranges for PAT attack.

Transformation	Min	Max
Initial camera x (m)	-1.5	1.5
Initial camera y (m)	-11.0	-6.0
Initial camera z (m)	0.6	1.8
Initial camera roll (°)	0.0	0.0
Initial camera pitch (°)	-5.0	5.0
Initial camera yaw (°)	-15.0	15.0
Camera Δx (m)	-0.1	0.1
Camera Δy (m)	-0.5	0.5
Camera Δz (m)	-0.1	0.1
Camera Δroll (°)	0.0	0.0
Camera Δpitch (°)	-3.0	3.0
Camera Δyaw (°)	-3.0	3.0
Initial target x (m)	-1.4	1.4
Initial target y (m)	-5.0	-0.7
Initial target z (m)	0.0	0.0
Initial target roll (°)	0.0	0.0
Initial target pitch (°)	0.0	0.0
Initial target yaw (°)	0.0	180.0
Target Δx (m)	-0.1	0.1
Target Δy (m)	-0.1	0.1
Target Δz (m)	0.0	0.0
Target Δroll (°)	0.0	0.0
Target Δpitch (°)	0.0	0.0
Target Δyaw (°)	-10.0	10.0
Lighting diffuse hue	0.0	360.0
Lighting diffuse saturation	0.0	0.2
Lighting diffuse value	0.1	0.7

Table 2: Discrete EOT variable selections for PAT attack.

Backgrounds	Targets
`school`	`green person`
`forest`	`PR2`

Appendix C. Trained GOTURN models

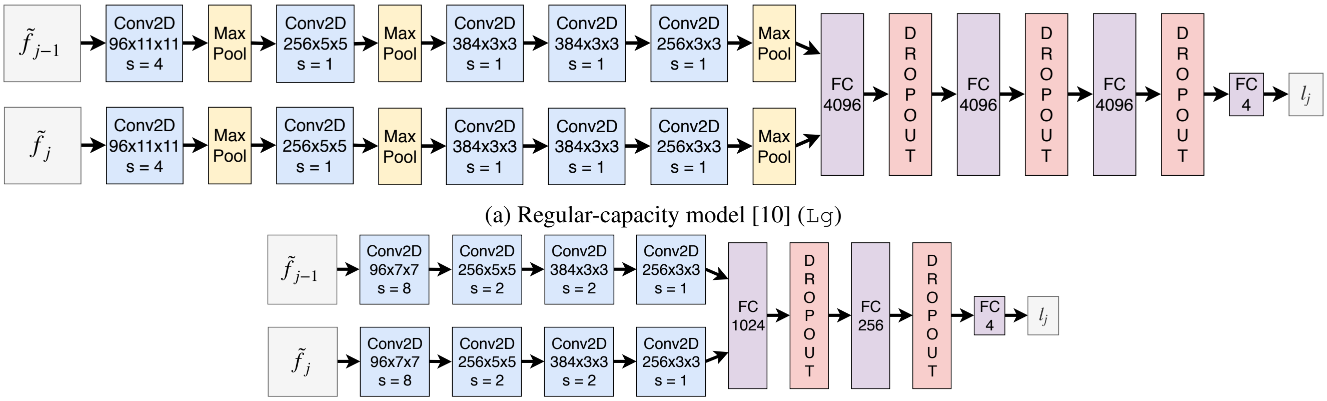
Figure 10 illustrates the two GOTURN neural object tracking architectures used in our experiments.

Appendix D. Baseline PAT Attack Settings

The parameters used in the baseline PAT attack settings (see Section 6.1.3) were determined using hyperparameters search, and from conducting sensitivity analyses on EOT minibatch size and iteration, as well as texture attributes experiments.

D.1. EOT Minibatch Size and Iteration

Similar to how training a neural network using Stochastic Gradient Descent (SGD) is sensitive to hyperparameter

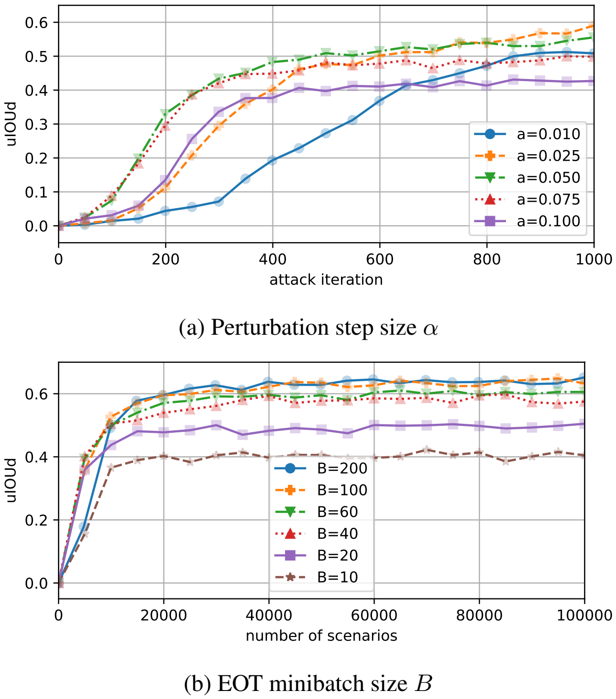


(a) Regular-capacity model [10] (`Lg`)

(b) Reduced-capacity model (`Sm`)

Figure 10: Neural architectures for the GOTURN object tracker instances.

settings, we analyzed the sensitivity of our proposed PAT attack method to its hyperparameters. We suspect that attacks using smaller EOT minibatch sizes B would require more iterations I to converge assuming a fixed perturbation step size α, while attacks using large minibatch sizes B would require an impractical amount of computing time per iteration. Thus, it is practically beneficial to balance the combination of the perturbation step size α and the minibatch size B, given a fixed number of attack iterations I.



(a) Perturbation step size α

(b) EOT minibatch size B

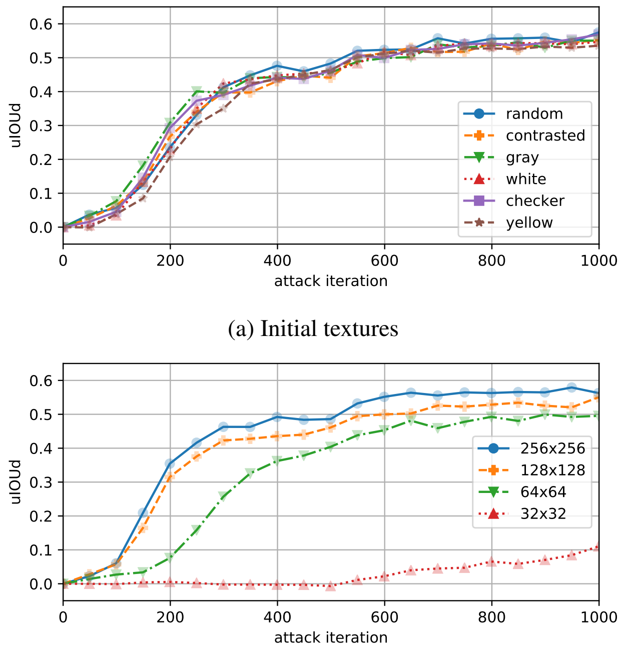
Figure 11: Adversarial strength over attack iterations, for various α values and EOT minibatch size.

We first optimized α for a fixed minibatch size of $B = 20$. As shown in Figure 11a, a step size of $\alpha = 0.025$ attained the best end-performance, however $\alpha = 0.075$ converged initially much faster. This trade-off substantiates our empirical observations and suggests that the source texture initially needs to have most of its pixels *broadly* perturbed to cause adversarial texture patterns to emerge, which would require drastic pixel changes with large perturbation sizes. Subsequently, however, slight *localized* pixel enhancements around "critical adversarial patterns" (see Section 6.4) steadily enhance the PAT's adversarial strength. Thus, we recommend a practical schedule that starts with a large perturbation size of $\alpha = 0.075$ for 500 attack iterations, and then refines using a smaller step size of $\alpha = 0.025$.

Next, using a single non-scheduled perturbation size of $\alpha = 0.075$, we varied the EOT minibatch size B. Note that, in Figure 11b, $\mu IOUd$ is plotted against the number of total EOT scenarios observed, i.e., $B \times I$. These results show consistent performance trends that are proportional to $B \times I$, i.e. the total number of scenes seen by each PAT attack, rather than the number of attack iterations I itself. Also, beyond small values of $B \geq 20$ that lead to high-variance stochastic gradient updates, larger minibatch sizes result in similar and *diminishing* amounts of improvement in both initial convergence speed and asymptotic adversarial strength. Consequently, we chose $B = 20$ for the best trade-off between compute per attack iteration and convergence.

D.2. Texture Attributes

Various related work made different recommendations on which source texture to use for best results. In particular, suggestions included all-white and all-yellow [27], and a *random contrasted* checkerboard pattern alternating between uniform sampling of $[0, 255]$ and $\{0, 255\}$ [26]. We also tried an all-gray source pattern, as well as a per-pixel randomly-sampled source.



(a) Initial textures

(b) Texture sizes

Figure 12: Adversarial strength among various initial textures and texture sizes.

However, as shown in Figure 12a, we found that initializing the texture with different patterns did not result in significant changes in convergence nor performance.

We also explored the effects of changing texture sizes and found that using a resolution of 32 × 32 lead to consistently poor results, while settings of 64 × 64, 128 × 128, and 256 × 256, yielded little differences in both initial convergence speed and asymptotic performance, as seen in Figure 12b. We thus chose 128 × 128 to balance between having sufficient pixel capacity to accommodate the wide ranges of EOT conditions, and amount of computation to compute texture perturbations. Still, we found it very important to be aware that our resolution choices are significantly affected by the viewing distances (see Appendix B) and poster sizes used in our experiments.

Appendix E. Ablation of EOT Conditioning Variables

In Section 6.3 of the main paper, we evaluated the effects of varying the ranges or choices for different EOT transformation variables, including background (-bg, +bg), target (-target, +target), lighting (-light, +light), poster size (small poster), camera pose (-cam pose, +cam pose), and target pose (-target pose, +target pose). Modified ranges to camera pose, target pose, and lighting are shown in Table 3, 4, and 5, respectively. Also, variations for (-bg, +bg) and (-target, +target) are as follows:

- -bg: use playground only;
- +bg: randomize among school, forest, playground and cafe;
- -target: use green person only;
- +target: randomize among green person, white person, t-shirt person, PR2 and robonaut.

Table 3: PAT attack settings for -cam pose and +cam pose.

Transformation	-cam pose		+cam pose	
	Min	Max	Min	Max
Initial x (m)	0.0	0.0	-2.0	2.0
Initial y (m)	-8.5	-8.5	-16.5	-5.5
Initial z (m)	1.2	1.2	0.4	2.2
Initial roll (°)	0.0	0.0	-1.5	1.5
Initial pitch (°)	0.0	0.0	-10.0	10.0
Initial yaw (°)	0.0	0.0	-20.0	20.0
Δx (m)	0.0	0.0	-0.15	0.15
Δy (m)	0.0	0.0	-0.80	0.80
Δz (m)	0.0	0.0	-0.15	0.15
Δroll (°)	0.0	0.0	0.0	0.0
Δpitch (°)	0.0	0.0	-5.0	5.0
Δyaw (°)	0.0	0.0	-5.0	5.0

Table 4: PAT attack settings for -target pose and +target pose.

Transformation	-target pose		+target pose	
	Min	Max	Min	Max
Initial x (m)	0.0	0.0	-1.6	1.6
Initial y (m)	-2.7	-2.7	-5.0	-0.7
Initial z (m)	0.0	0.0	0.0	0.0
Initial roll (°)	0.0	0.0	0.0	0.0
Initial pitch (°)	0.0	0.0	0.0	0.0
Initial yaw (°)	90.0	90.0	-90.0	270.0
Δx (m)	0.0	0.0	-0.15	0.15
Δy (m)	0.0	0.0	-0.15	0.15
Δz (m)	0.0	0.0	0.0	0.0
Δroll (°)	0.0	0.0	0.0	0.0
Δpitch (°)	0.0	0.0	0.0	0.0
Δyaw (°)	0.0	0.0	-20.0	20.0

Appendix F. Imitation Attacks

In Section 6.4, we set the value of $w_{ps} = 0.6$. This value was determined based on an experiment where we stud-

Table 5: PAT attack settings for -light and +light.

Diffuse Light Source	-light		+light	
	Min	Max	Min	Max
Hue	0.0	360.0	0.0	360.0
Saturation	0.0	0.0	0.0	0.7
Value	0.7	0.7	0.0	0.7

ied the effect of changing w_{ps} on the adversarial strength $\mu IOUd$ and perceptual similarity (as measured by the Euclidean L_2 distance to the source image in RGB colorspace). Unsurprisingly, as seen in Figure 13, smaller values of w_{ps} imposed fewer constraints and thus lead to faster attack convergence and better end-performance, while the inverse was true for larger values of w_{ps}. We thus chose $w_{ps} = 0.6$ after manually assessing which PATs had recognizable levels of perceptual similarity to their source images, as seen in Figure 18.

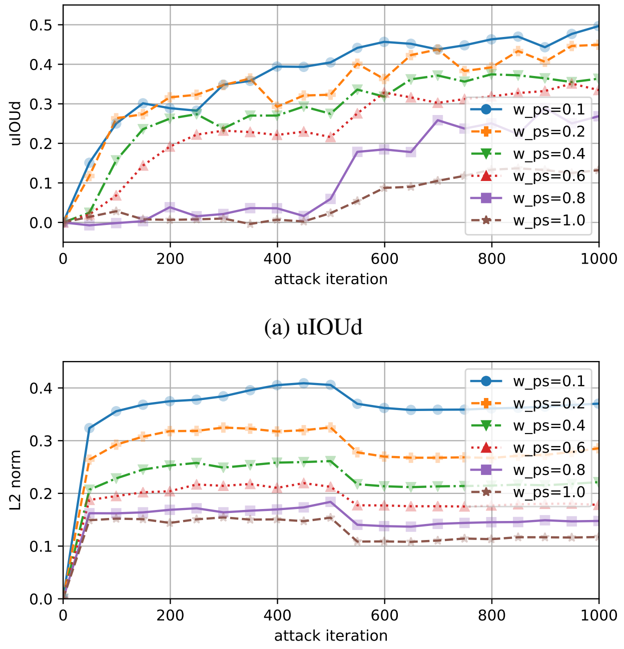
To substantiate Figure 6, Figure 14 illustrates how $\mu IOUd$ and perceptual similarity metrics change over attack iterations. As we can see, some specific combinations of initial posters and losses made the attack easier to converge. For example, performing attacks using waves as initial texture with hybrid losses ($\mathcal{L}_{nt}$ & $\mathcal{L}_{ga+}$) resulted in strong adversaries, while using non-targeted loss alone did not.

Figure 15 compares perceptual similarity ($L_2\,norm$) against adversarial strength ($\mu IOUd$) among $\mathcal{L}_{ga-}$, $\mathcal{L}_{nt}$, and $\mathcal{L}_{t-}$, and shows that, for a given threshold on $L_2\,norm$, $\mathcal{L}_{ga-}$ generally had better early convergence, but likely weakened adversarial strength after more attack iterations.

Figure 19 illustrates the emergence of "critical adversarial patterns" that we discussed in Section 6.4 in the main paper. In Figure 19a, the *critical* dark striped pattern started to emerge at around iteration 400, followed by the appearances of other nearby colorful patterns, which presumably were to drive predictions towards the central adversarial striped pattern. In contrast, when we imposed a perceptual similarity loss during an imitation attack, only the dark striped pattern eventually emerged after significantly more attack iterations, as seen in Figure 19b.

Appendix G. Transfer among tracking models

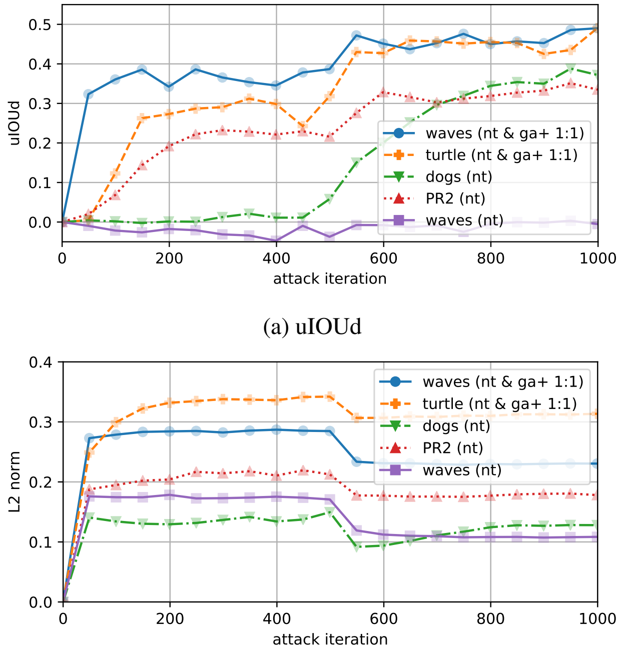
We evaluated the transferability of PATs among different tracking models. When evaluating PATs on GOTURN models trained using different datasets, the off-diagonal results in Figure 16a generally show that a decent-to-great amount of adversarial strength is still present. Nevertheless, we see that the transferred efficacy of adversaries varied based on the tracker model and the loss used. For instance, a sim-trained PAT optimized using $\mathcal{L}_{nt}$ and applied to the s2r GOTURN tracker is strongly adversarial, whereas a similar



(a) uIOUd

(b) L_2-norm perceptual similarity

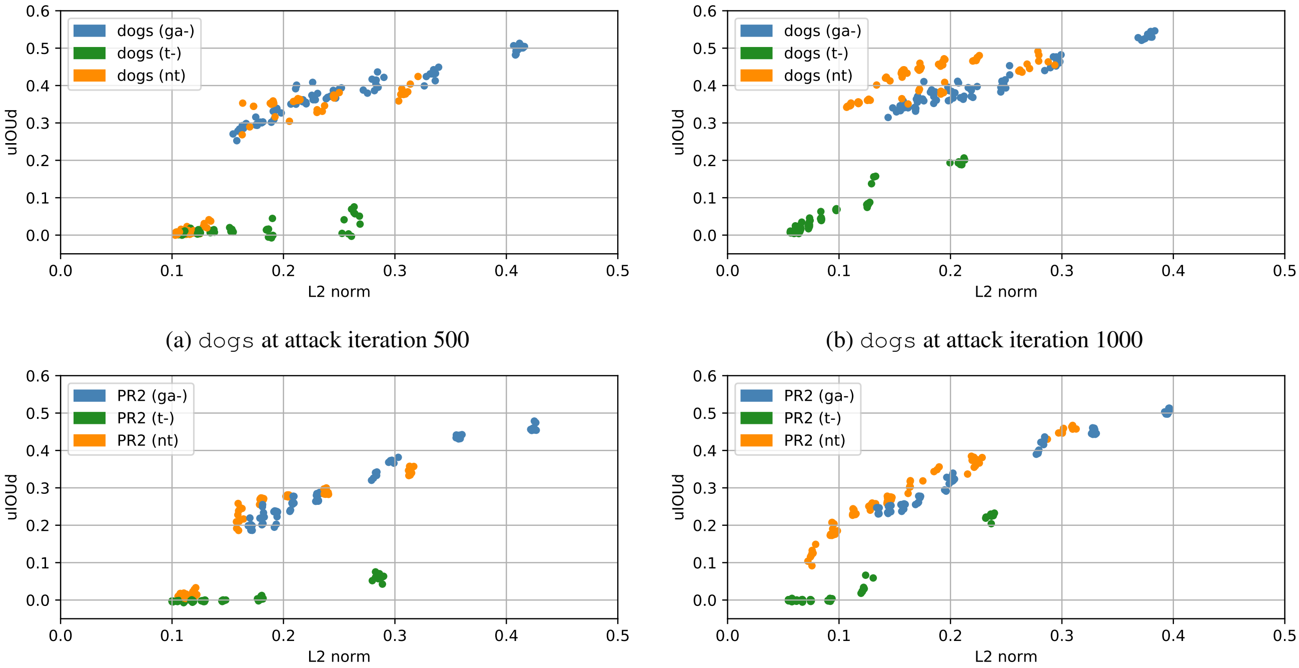
Figure 13: Adversarial strength and perceptual similarity among various w_{ps}.



(a) uIOUd

(b) L_2-norm perceptual similarity

Figure 14: Adversarial strength and perceptual similarity among source textures shown in Figure 6 of main paper.



(a) `dogs` at attack iteration 500

(b) `dogs` at attack iteration 1000

(c) `PR2` at attack iteration 500

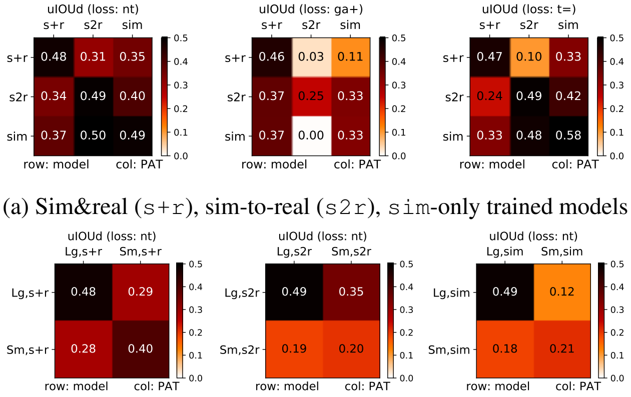
(d) `PR2` at attack iteration 1000

Figure 15: Performance of $\mathcal{L}_{nt}$, $\mathcal{L}_{ga-}$, and $\mathcal{L}_{t-}$ in imitating `dogs` and `PR2` textures for $w_{ps} \in [0.1 : 0.1 : 0.8]$.

PAT optimized using $\mathcal{L}_{ga+}$ becomes completely inert.

Similarly, PATs preserved some of their adversarial strength when transferred between trackers with different capacities, as seen in Figure 16b. However, while all PATs applied to reduced-capacity models (`Sm`) affected GOTURN predictions, their $\mu IOUd$ values around 0.20 do not reflect strong adversaries, thus indicating that it is more difficult to fool small-capacity GOTURN networks into consistently breaking away from their intended target.

Figure 17 shows the PATs used in this experiment. Generally, we observe similar adversarial patterns emerging from PAT Attacks on GOTURN models trained on different datasets, as well as different capacities, which explain why PATs transfer to a certain degree among different GOTURN trackers. The sole exception is seen from the second row of Figure 17a, which reflected the fact that the adversarial loss $\mathcal{L}_{ga+}$ caused different patterns to emerge for different models, albeit with similar levels of competent adversarial strength.



(a) Sim&real (`s+r`), sim-to-real (`s2r`), `sim`-only trained models

(b) Models with default (`Lg`) and reduced (`Sm`) capacities

Figure 16: Adversarial strength of generated PATs (columns) applied to different GOTURN tracking models (rows).

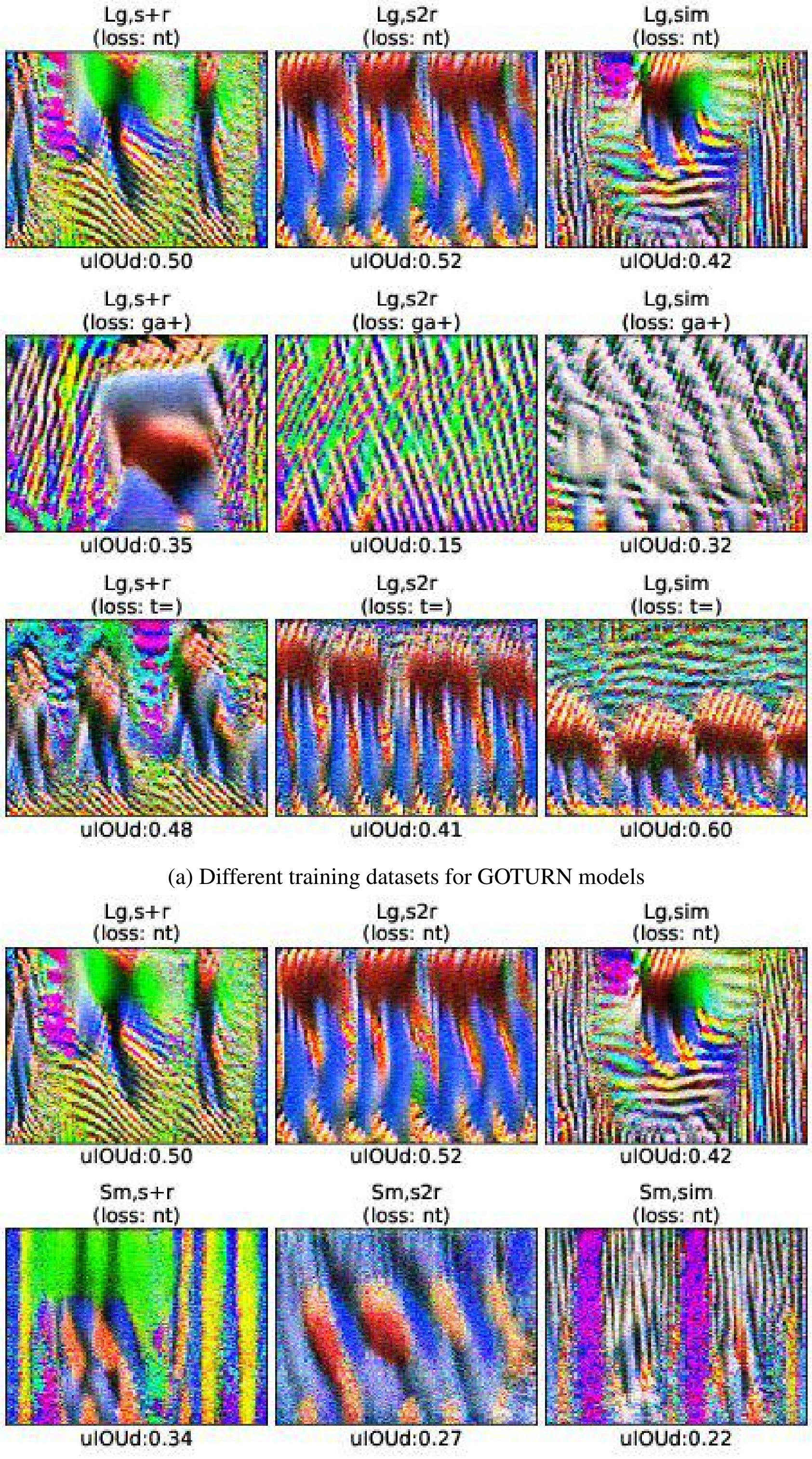
Appendix H. Demonstration of Sim-to-real Transfer

As discussed in Section 6.5, we conducted test runs in real-world tracking and servoing conditions, and qualitatively verified the transferred adversarial strength of our synthetically-generated PATs, especially those containing "critical adversarial patterns". Many of these real-world runs are shown in the supplementary video. Nevertheless, it is generally difficult to quantify performance consistently in the real world, due to tediousness and impracticality in labeling performance, controlling for repeated conditions, and dealing with practical complexities such as limited battery life and hardware failures. Still, we segmented runs into video clips, and manually labeled them as either `strong`ly adversarial (i.e. where the tracker jumps onto the PAT and stays locked onto it even when momentarily obstructed), `weak`ly adversarial (i.e. where the tracker sometimes switches from the person to the PAT, and tends to latch back onto the person), or `fail`ure.

Looking at Table 6, we see that the tracker was quickly drawn to PATs when deployed on a stationary camera. On the other hand, it was much harder to fool the person tracker when the drone was servoing the target. Whether the PAT was displayed digitally on a monitor, or printed as an `A0` poster, we anecdotally observed that both of these materials displayed some amount of specular reflections. These specularities changed as the camera moved around, and thus likely had altered the appearances of PATs during our servoing runs and rendered them inert. Therefore, devising adversaries that are robust to specularities would be an exciting avenue for future research.

Table 6: Physical-world attack performance.

Runs	Strong	Weak	Fail
Stationary	57 (71%)	13 (16%)	10 (13%)
Servo	6 (33%)	5 (28%)	7 (39%)



(a) Different training datasets for GOTURN models

(b) Different network capacities for GOTURN models

Figure 17: PATs used in the transferability among tracking models experiment.

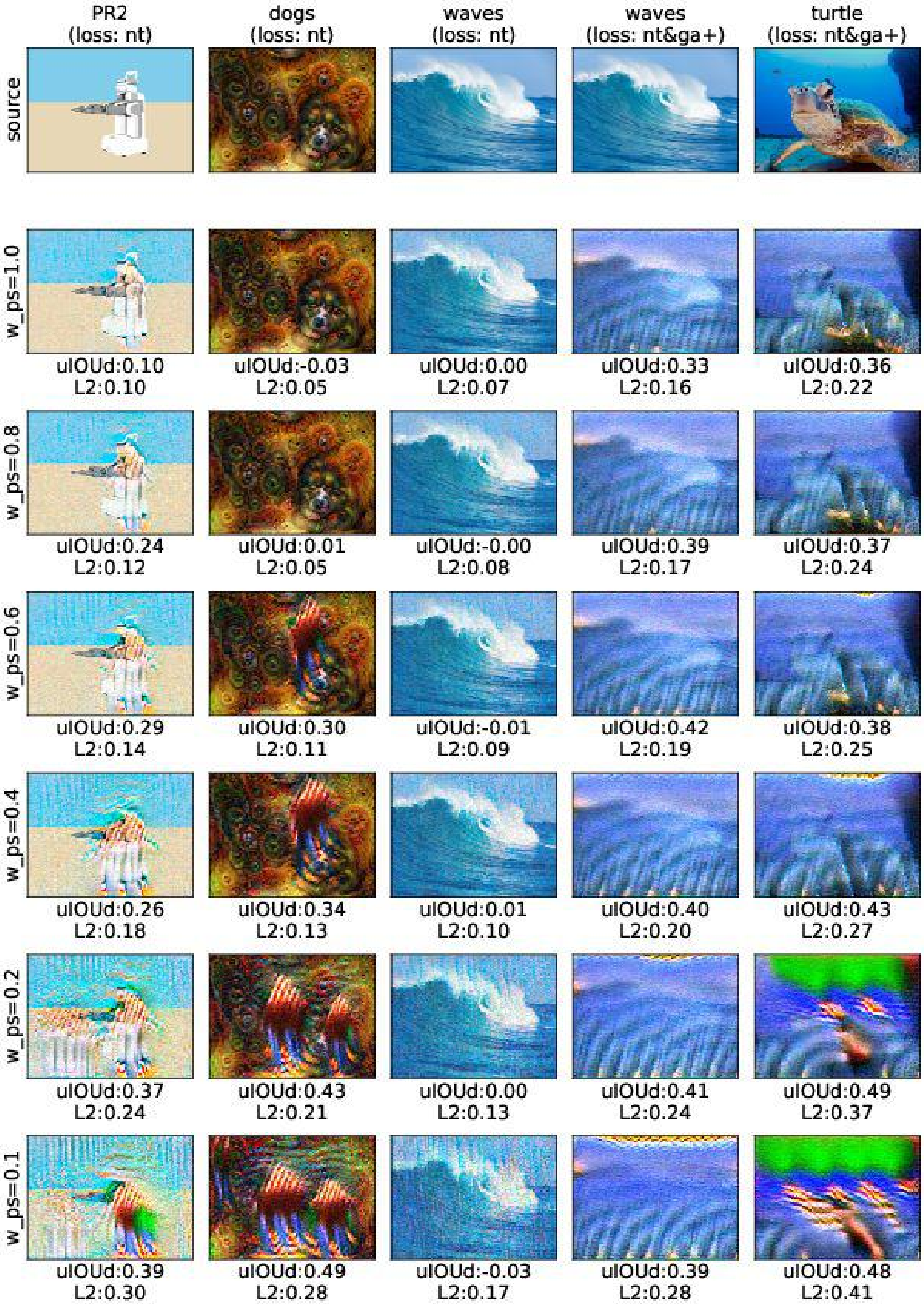
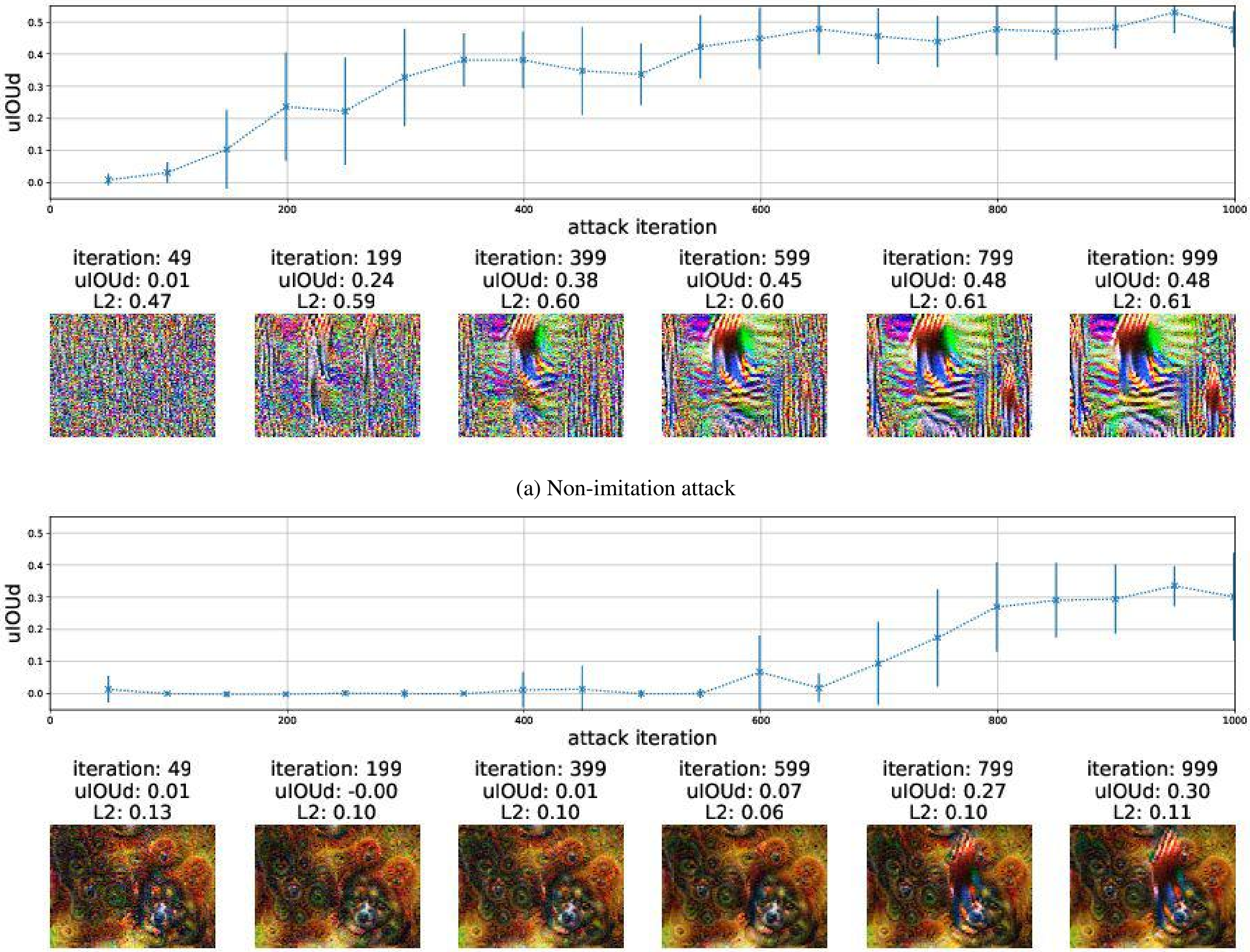


Figure 18: PATs generated with various w_{ps} values.



(a) Non-imitation attack

(b) Imitation attack

Figure 19: The emergence of "critical adversarial patterns" for non-imitation and imitation attacks.

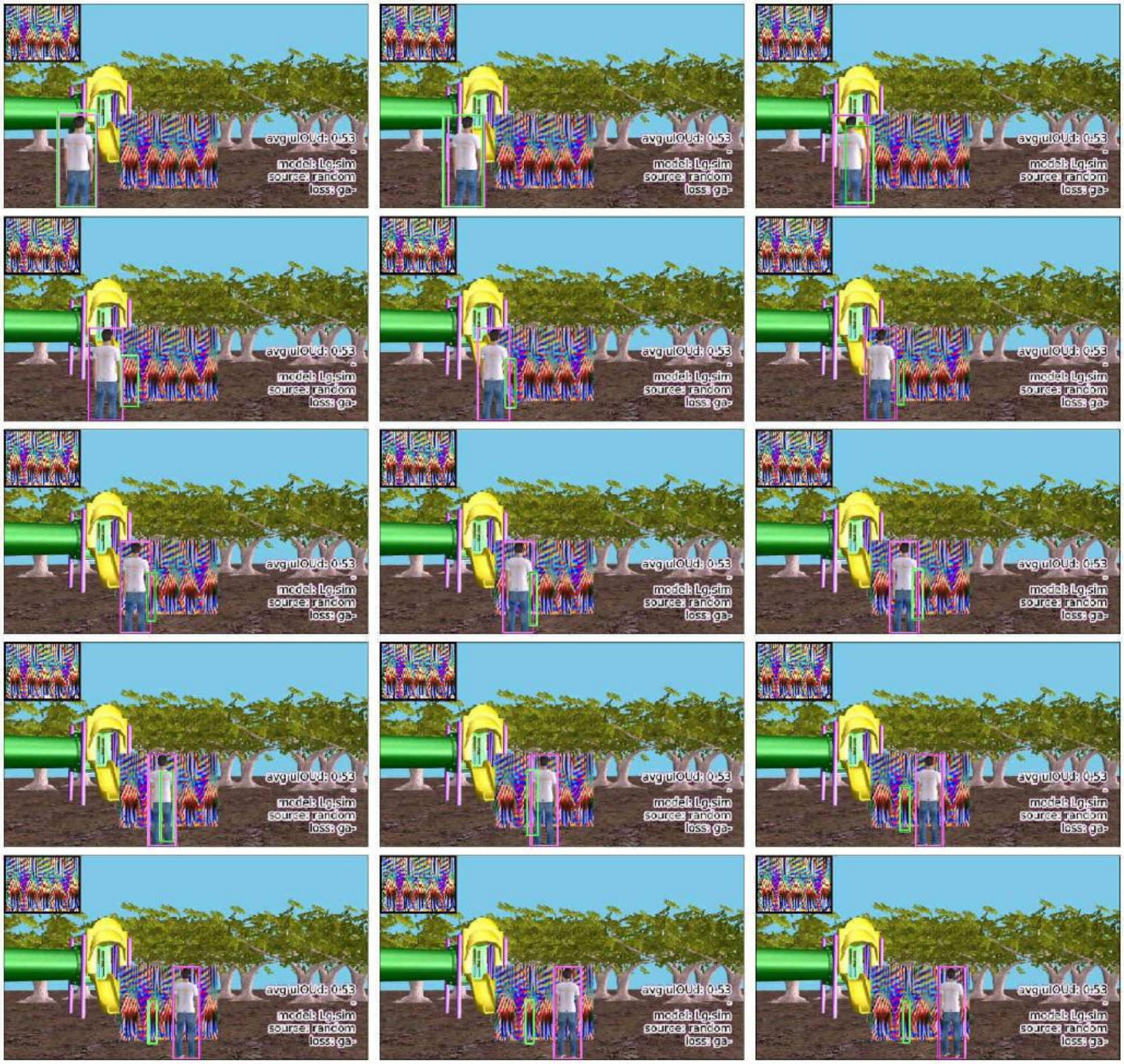


Figure 20: PAT fools the tracker in simulation. Here, the purple bounding box represents the ground truth bounding box of the tracked object, while the green bounding box represents the tracker's prediction. Note that the sequence starts from the top-left frame to the bottom-right frame.

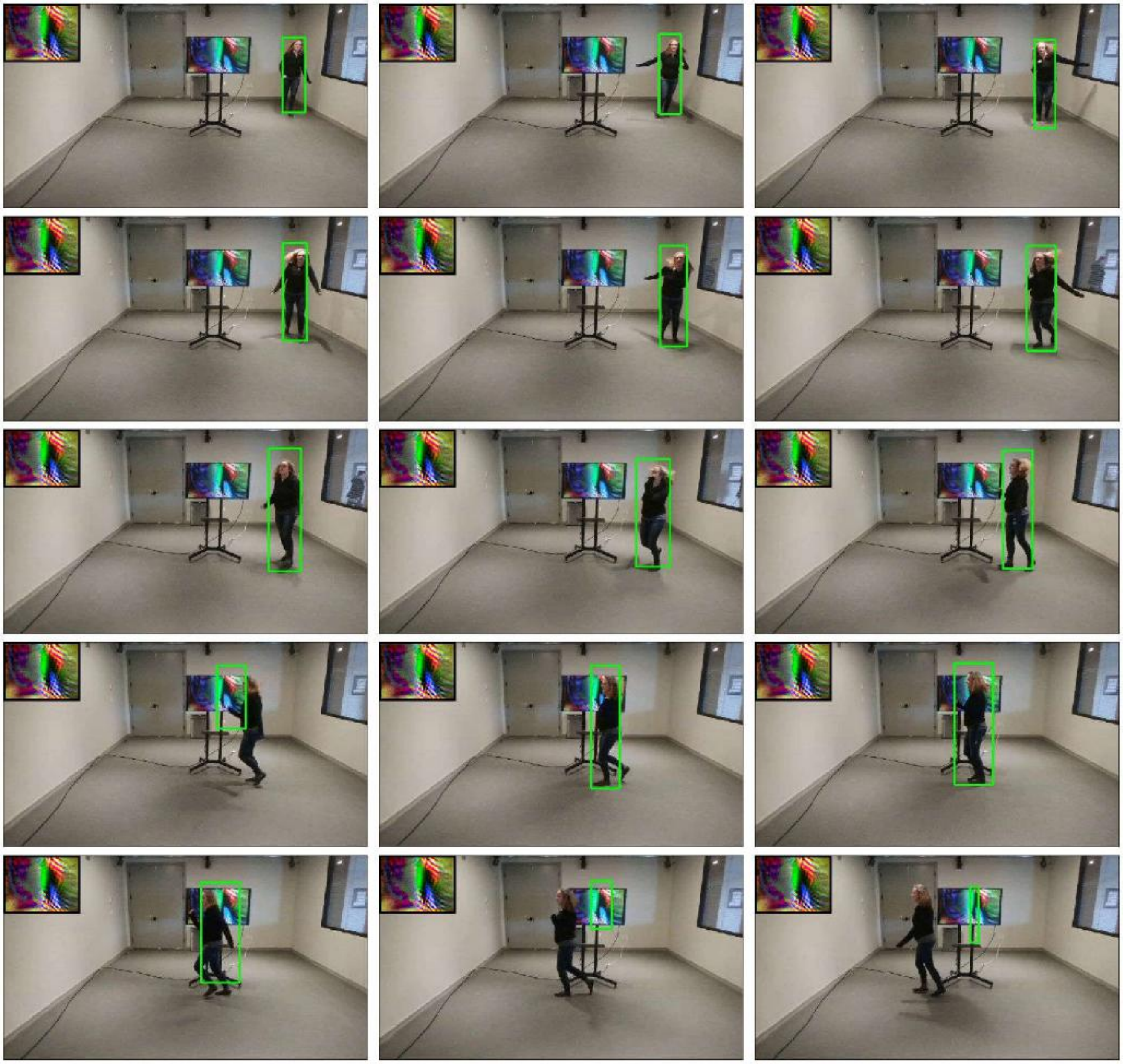


Figure 21: PAT fools the tracker in the real world indoor setting, where the PAT is displayed on a TV. Note that the sequence starts from the top-left frame to the bottom-right frame.

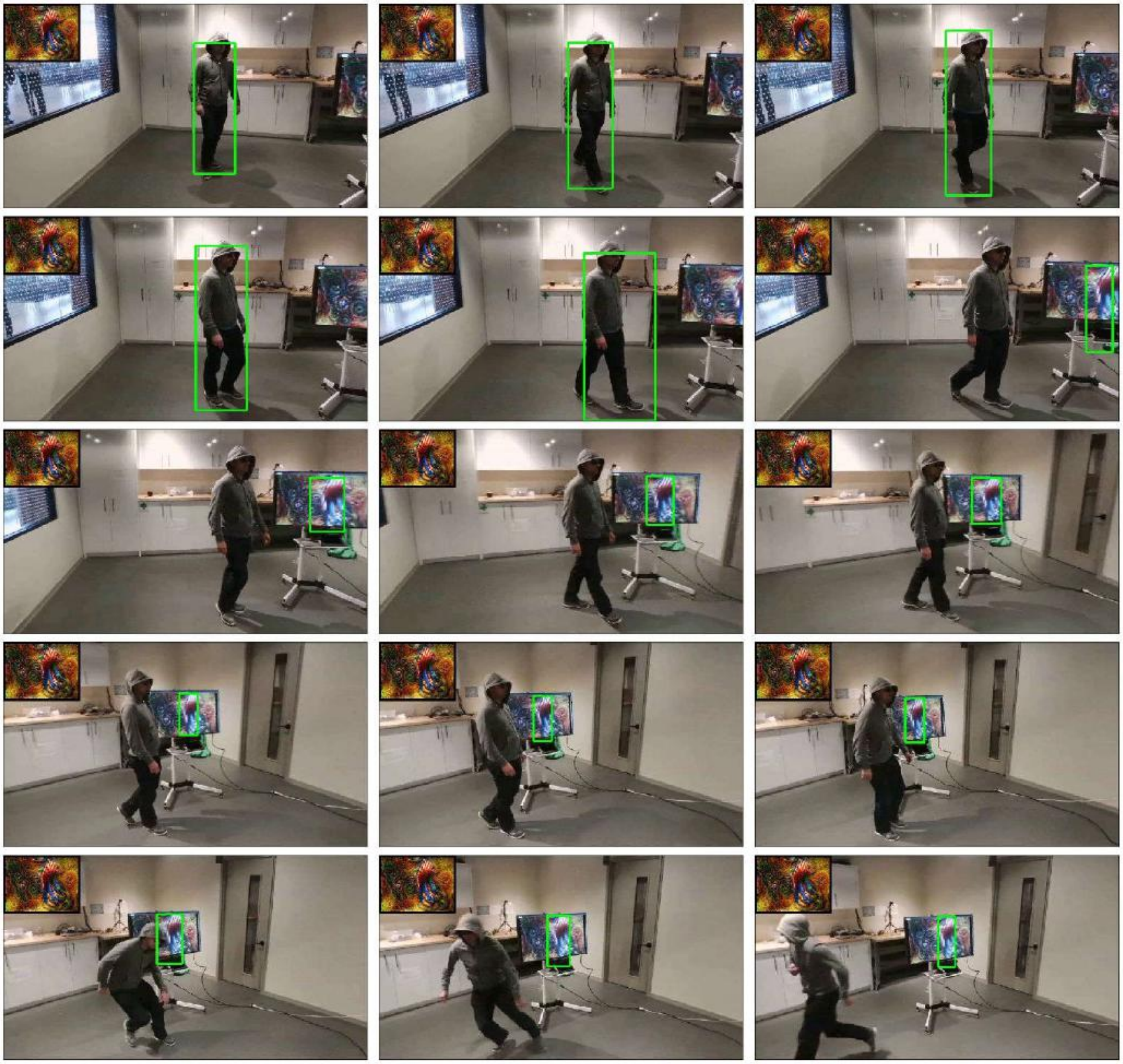


Figure 22: PAT fools the tracker in the real world indoor setting during servoing run. Note that the sequence starts from the top-left frame to the bottom-right frame.

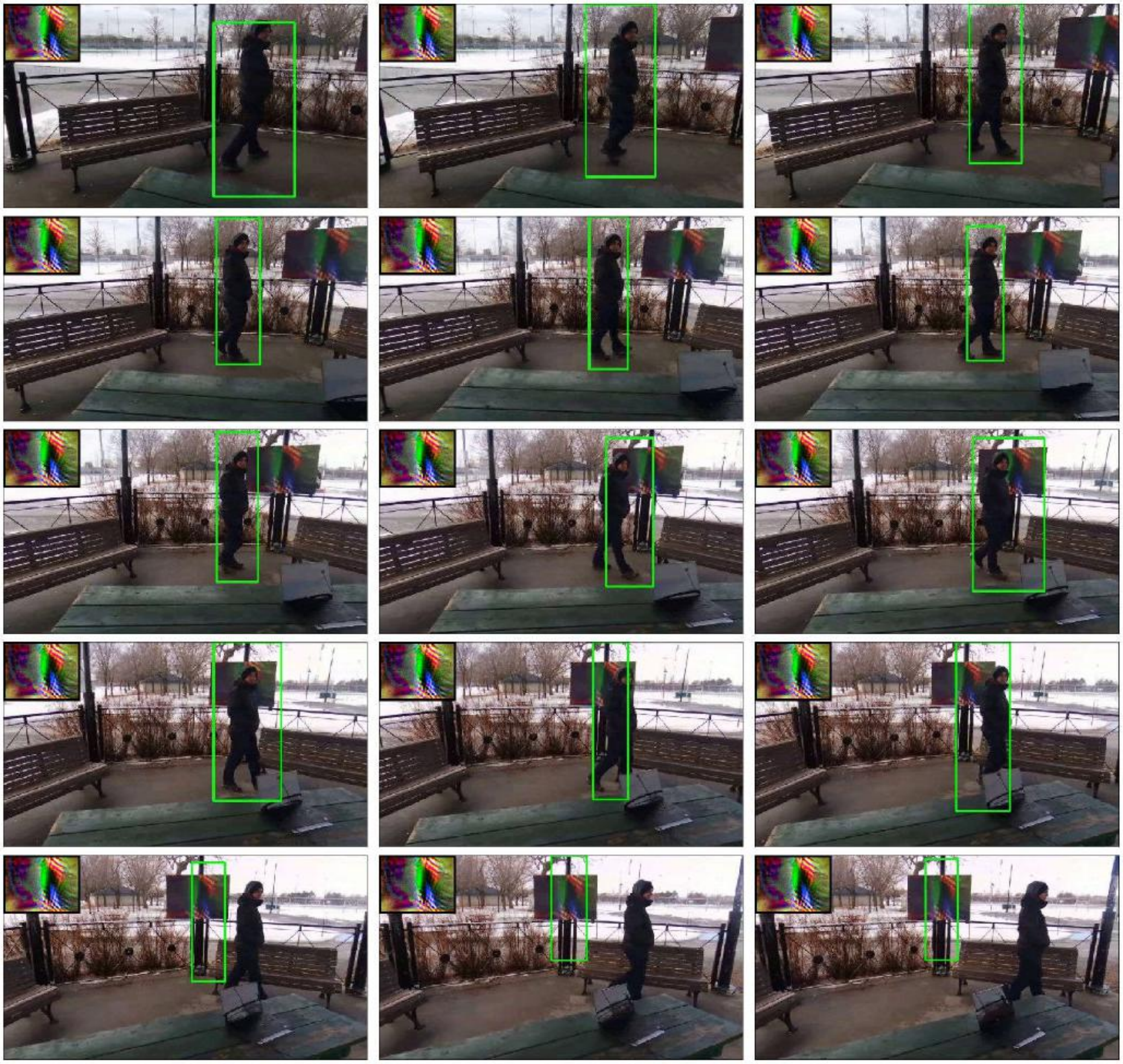


Figure 23: PAT fools the tracker in the real world outdoor setting during servoing run, where the PAT is printed as a poster. Note that the sequence starts from the top-left frame to the bottom-right frame.