Controlling Autonomous Vehicles using Continuous Adversarial Patches

Pia Hanfeld
BIFOLD & TU Berlin
Berlin, Germany

Erik Imgrund
BIFOLD & TU Berlin
Berlin, Germany

Felix Weißberg
BIFOLD & TU Berlin
Berlin, Germany

Thorsten Eisenhofer
CISPA Helmholtz Center for Information Security
Saarbrücken, Germany

Wolfgang Hönig
TU Berlin
Berlin, Germany

Konrad Rieck
BIFOLD & TU Berlin
Berlin, Germany

Abstract

Autonomous vehicles, including self-driving cars and drones, heavily rely on machine learning models for perception and control. This reliance introduces a significant attack surface through adversarial patches—visual patterns that mislead learning models. While prior work has demonstrated that such patches can induce misclassifications or tracking failures, they fall short of enabling sustained control over autonomous systems in rapidly changing conditions.

In this paper, we introduce continuous adversarial patches, a proof-of-concept attack that demonstrates the potential for sustained control over autonomous vehicles that rely on single object tracking. When displayed on a video screen, these dynamic patches guide tracking drones along adversary-defined trajectories. We implement a diffusion model to produce continuous adversarial patches in real time, targeting PULP-Frontnet and YOLOv5, two object detection models used in drone research. In simulation, our method demonstrates successful trajectory manipulation under controlled conditions, including slingshot and freestyle maneuvers. We release our simulation environment and code to facilitate further investigation of this attack class and the development of robust countermeasures. Our code is available for further research under github.com/mlsec-group/continuous-patches.

CCS Concepts

• **Computing methodologies** → **Machine learning**; *Vision for robotics*; • **Security and privacy** → **Software and application security**.

Keywords

Adversarial Machine Learning, Autonomous Vehicles, Security

ACM Reference Format:
Pia Hanfeld, Erik Imgrund, Felix Weißberg, Thorsten Eisenhofer, Wolfgang Hönig, and Konrad Rieck. 2026. Controlling Autonomous Vehicles using Continuous Adversarial Patches. In *19th Workshop on Artificial Intelligence and Security (AISec '26), November 15–19, 2026, The Hague, Netherlands.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3847352.3848104

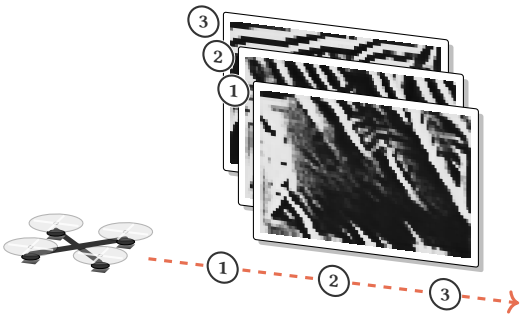


Figure 1: A person-following drone is accelerated along an adversary-defined trajectory via continuous adversarial patches presented on a video display.

1 Introduction

Autonomous vehicles, from self-driving cars to aerial drones, heavily rely on machine learning for perception and control [19, 34, 38]. Yet, these same models introduce a critical vulnerability in the form of adversarial patches. These visual patterns can trick systems into misclassifying objects or losing track of targets [2, 13, 33, 43, 46, 53]. However, all existing attacks have been limited to instantaneous errors. That is, the adversary predefines a single target action and crafts a static visual pattern accordingly. The adversarial pattern cannot be changed in real time to adapt to environmental changes. As the vehicle changes position and orientation, the visible patch region shifts, the viewing angle changes, and a static pattern quickly loses efficacy. Controlling a vehicle's *trajectory* thus requires continuously generating new patches, each adapted to the vehicle's current state.

This requirement poses three distinct technical challenges. First, the attacker does not have access to the victim's camera image and therefore cannot optimize patches against the actual scene the vehicle perceives. Second, standard gradient-based optimization methods, such as projected gradient descent (PGD), are computationally prohibitive for real-time use: generating a single effective patch requires seconds to minutes [2, 13, 31], whereas a moving drone operating at a 10 Hz control loop demands patch updates at comparable frequency. Third, the patch must remain effective across a wide range of vehicle poses, display positions, and target actions, not just for a single fixed configuration. Precomputing patches for

AISec '26, The Hague, Netherlands

ACM ISBN 979-8-4007-3031-3/2026/11
https://doi.org/10.1145/3847352.3848104

all possible states is intractable, as the joint space of vehicle poses and display constraints is continuous and high-dimensional.

We bridge this gap by introducing *continuous adversarial patches*, a proof-of-concept attack class that explores whether video displays can serve as real-time steering mechanisms. Unlike static patches, our method employs a generative model conditioned on the vehicle's state to produce adaptive, time-varying patterns. When displayed on a screen, these dynamic patches can guide tracking drones along adversary-defined trajectories in a simulated environment, as shown in Fig. 1.

We evaluate our approach using a custom simulation environment and a generative model to compute adversarial patches. In this controlled setting, our attack demonstrates the ability to deceive PULP-Frontnet and YOLOv5, two object detection models studied in drone research, inducing maneuvers such as "slingshot" flights where a drone is aggressively accelerated along attacker defined directions. This proof-of-concept shows that control over a part of the field of view can be transformed into an active attack vector, with the potential to influence a tracked vehicle's trajectory through a nearby display.

Our findings suggest that continuous adversarial patches represent a previously underexplored attack class warranting further investigation, particularly in public spaces with large displays (e.g., festivals, sports events).
In summary, we make the following contributions.

- *Continuous Patches:* We introduce a proof-of-concept attack that generates adaptive patches in real time using a diffusion model, demonstrating sustained influence over autonomous vehicles relying on object tracking for navigation.
- *Simulation Environment:* We design a custom simulation environment consisting of a drone's object detector and control dynamics, released as open-source software. It evaluates patches under controlled conditions and can serve as an adversarial benchmark for future work.
- *Generative Training:* We introduce a training objective for diffusion models that integrates the victim's detection pipeline, incorporating the distribution of real-time transformations and environmental states rather than optimizing on static images only.
- *Proof-of-Concept Evaluation:* We demonstrate the effectiveness of our approach under varying conditions in simulation, including the ability to induce slingshot maneuvers and complex trajectories against targeted detection models.

2 Attacks on Autonomous Vehicles

Autonomous vehicles, such as aerial drones, self-driving cars, and ground robots, operate without direct human input. They can navigate complex environments and pursue specific objectives on their own. For example, consumer aerial drones can autonomously track and film a moving person, enabling the creation of personal video content. Similar capabilities are also used in self-driving cars for lane monitoring and in ground robots for navigating around obstacles [30, 39].

2.1 Object Detection and Tracking

Central to autonomous vehicles is their ability to perceive and understand their surroundings. To this end, a system must be able to detect and track nearby objects. This task, referred to as *object detection and tracking*, involves estimating the position and orientation of an object—its *pose*—relative to the vehicle and continuously updating this estimate as both the vehicle and the object move.

For object detection, most autonomous vehicles rely on onboard cameras. The captured images are processed by machine learning models that identify relevant objects and estimate their pose within a predefined coordinate system. In practice, objects are first represented by bounding boxes in the camera images, annotated with a likelihood of which object class the box belongs to. Finally, the boxes are filtered and mapped to physical poses in the vehicle's frame of reference. Object tracking extends this process across successive video frames by using re-identification algorithms that compare visual features extracted from bounding boxes and assign consistent identifiers as objects move through the scene.

Two architectural paradigms exist in practice. End-to-end systems like PULP-Frontnet [34] directly predict poses from camera images, avoiding intermediate representations. Alternatively, systems may decompose the task: an object detector first localizes candidates via bounding boxes and an object tracker re-identifies the same object from previous frames based on learned features [9, 24]. Initialization of object trackers typically requires an initial detector output, with YOLO variants commonly used in this stage [53]. For environments with limited scene complexity (e.g., single-person tracking in controlled backgrounds), detector-only pipelines can suffice since filtering disambiguates the tracked object. This design choice trades multi-target robustness for computational efficiency, which is advantageous on resource-constrained platforms.

2.2 Adversarial Patches

For many years, machine learning has defined the state of the art for object detection and tracking, with deep learning models excelling in performance and efficiency. Unfortunately, these models also introduce a severe vulnerability for autonomous vehicles, particularly in the form of adversarial patches. Such patches are carefully crafted visual patterns that, when placed within the field of view of a vehicle's camera, mislead the learning model and cause erroneous predictions. This misdirection then propagates through the perception and control pipeline of the vehicle and directly influences its behavior.

A substantial body of prior work has demonstrated attacks against self-driving cars [3, 5, 45, 46, 54] and drones [13, 33, 44, 53] using adversarial patches. These attacks, for example, trick models into misclassifying traffic signs, ignoring pedestrians, or causing objects to simply vanish during tracking [5, 45, 54]. DeGhost [33] and FlyTrap [53] both establish the feasibility of visually deceiving drones, but neither achieves the fine-grained, trajectory-level control enabled by state-conditioned continuous patches (see Section 8 for a detailed comparison). However, most previous attacks miss a key aspect of autonomous vehicles: they are not *static*. While autonomous vehicles naturally move and navigate, existing attacks are limited to a fixed instantaneous action and are therefore unable to dynamically influence the behavior of the victim.

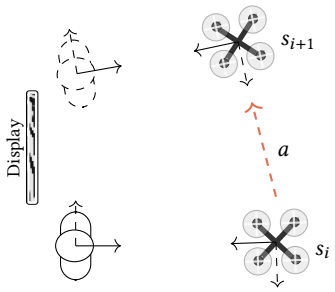


Figure 2: Conceptual description of the attack scenario. At state s_i, the drone aims to track a human subject. A continuous adversarial patch on a display induces a false detection (dashed), triggering the action a and leading to the state s_{i+1}.

3 Continuous Adversarial Patches

We introduce a new attack that explicitly addresses the dynamic nature of autonomous vehicles. The key idea is to construct continuously adversarial patches that are adapted to the vehicle's current state and thereby dynamically influence its motion. Rather than being a static pattern, the patch is a function conditioned on the desired state of the autonomous vehicle. Before introducing the design of this function, however, we first define our threat model, which naturally differs from that of classic adversarial patches.

3.1 Threat Model

We consider an adversary whose goal is to manipulate the trajectory of an autonomous vehicle to cause physical harm, for example by steering it toward bystanders. The vehicle derives waypoints for its navigation from poses estimated by an internal object tracking model. The adversary seeks to influence this navigation by manipulating the visual inputs to the model. This setup reflects common deployment scenarios in which object-tracking outputs serve as a primary input to control, for example in consumer drones that follow people, as produced by companies such as DJI (e.g., the Neo), Bitcraze (e.g., Crazyflie 2.1), and HoverAir (e.g., the X1).

We assume that the adversary controls a *video display* within the vehicle's field of view and can estimate the vehicle's pose. Suitable displays include monitors mounted on trucks for advertising at public events, LED walls used for image magnification at concerts, and large displays installed along city boulevards. Fig. 3 shows examples of displays that fall within our threat model. Although presenting adversarial patches via a display requires more effort than using stickers or murals, such video screens are common in urban and public settings, such as festivals, stadiums, and tourist areas.

Estimating the pose of an object is a well known problem in computer vision and robotics. An extensive amount of prior work has enabled six degrees of freedom pose estimation of objects from a single RGB image in real time [8, 17, 18, 36]. We therefore assume that the attacker is able to perform the detection, tracking and pose estimation of UAVs in the vicinity of one or multiple smartphones [22].

Finally, we assume the attacker possesses white-box access to the detection model employed by the vehicle. This assumption is

(a) Food truck with LED displays

(b) Billboard truck with LED displays

Figure 3: Examples of video displays at public events. Generated with Gemini 3.1 Pro.

motivated by the prevalence of publicly available and standardized object detection models in modern autonomous systems [6, 51, 53]. Even for proprietary systems, transfer attacks using surrogate models are feasible [35]. This threat model intentionally represents a worst-case adversary designed to probe the upper bound of vulnerability. We discuss the implications of relaxing these assumptions in Section 7.

3.2 Problem Statement

If an adversarial patch is a function rather than a point, how do we describe its generation? To address this question, we first model the motion of the victim system and then derive optimization problems that yield a corresponding function.

State Model of Attack. We represent the movement of the victim by a series of pose states $s_0, \ldots, s_n$. In our setting, the orientation is especially important, as it allows the attacker to accurately determine the field of view of the camera. Although the pitch and roll of some vehicles may also vary, we can omit them, as cameras are typically mounted on gimbals that stabilize these axes. We therefore represent the orientation using only the yaw angle. This results in a pose representation $s \in \mathbb{S} = \mathbb{R}^4$, where the first three dimensions describe the victim vehicle's position in world coordinates, while the last dimension encodes the yaw angle. We further denote the distance between two poses by $\|s_a - s_b\|$ and calculate it as the sum of the L_2-distance of the position and the cosine distance of the yaw angle.

Based on this representation, we model the state transitions as $s_{t+1} = s_t + a_t$, where the action $a \in \mathbb{A} = \mathbb{R}^4$ represents a relative change in a pose. Note that this operation is a vector addition only in the first three dimensions and an addition modulo 2π in the last. In a person-following drone, the action typically depends only on the camera image $x_t \in \mathbb{X}$ and is computed using an object tracking system $\gamma : \mathbb{X} \to \mathbb{A}$, where $\mathbb{X} = \mathbb{R}^{h_x \times w_x}$ is the set of grayscale camera images with height h_x and width w_x.

While the attacker cannot directly control the input image x_t, they can influence a constrained region of it. These constraints arise from the position of the video display on which the patch is shown relative to the vehicle's field of view. This constrained region can be described by a mask $c \in \mathbb{C} = \{0,1\}^{h_x \times w_x}$. The attacker generates patches that are shown in this constrained region using a *patch generation function* $p : \mathbb{A} \times \mathbb{C} \to \mathbb{P}$. The possible patches $\mathbb{P} = \mathbb{R}^{h_p \times w_p}$ have a fixed size and must be transformed to fit in the constrained region. We denote the addition of such a transformed patch to a camera image x_t under constraints c by

$$x_t \oplus_c p(\cdot, \cdot). \tag{1}$$

Patch Size and Perspective. The possible patches $\mathbb{P} = \mathbb{R}^{h_p \times w_p}$ have fixed resolution and must be transformed to fit in the constrained region. While the physical size of the patch's projection onto the camera image varies with the drone's distance to the display, we decouple patch generation from physical geometry. Perspective transformations (see Section 4) map the fixed-resolution patch into the constrained region at render time. This allows the diffusion model to learn a resolution-invariant representation while accommodating varying display sizes and distances during evaluation.

Adversarial Trajectories. The attacker can now control the victim's state *trajectory*. While a single patch induces only a single state change, a continuous patch generates a sequence of actions and thus gradually shapes the vehicle's trajectory over time. We focus on two types of such trajectories: *slingshot* maneuvers, which aim to aggressively accelerate the vehicle in a target direction in order to cause a crash, and *freestyle* maneuvers, in which the resulting trajectory can take arbitrary forms.

Slingshots: We model the attacker's goal of a slingshot as moving towards a goal point $g \in \mathbb{S}$ far away in the target direction. The attacker constructs this attack by finding a patch generation function p based on the current constraints and the target action $(g - s_t)$. More formally, the attacker aims to solve the optimization problem

$$\underset{p}{\operatorname{argmin}} \, \|g - (s_t + a_t)\| \tag{2}$$

$$\text{with} \;\; a_t = \gamma(x_t \oplus_{c_t} p(g - s_t, c_t)) \,. \tag{3}$$

Freestyle: In this scenario, we model the attack as a sequence of k goal points $(g_1, \ldots, g_k)$ that define a target trajectory. The attack is realized by learning a patch generation function p that, at each time step t, induces actions that keep the vehicle close to the current goal point. Concretely, the objective is to minimize the deviation $(g_i - s_t)$. This leads to the following optimization problem,

$$\underset{p}{\operatorname{argmin}} \sum_{i=0}^{k} \|g_i - (s_t + a_t)\| \tag{4}$$

$$\text{with} \;\; a_t = \gamma(x_t \oplus_{c_t} p(g_i - s_t, c_t)) \,. \tag{5}$$

3.3 Attack Methodology

The patch generating function p can be implemented in various ways. A naive approach would be to iteratively apply a standard algorithm for generating adversarial patches. In the setting of controlling an autonomous vehicle, however, specific challenges arise that render the direct application of existing techniques ineffective. First, most approaches to adversarial patch generation require access to the camera image on which the patch is placed, which is not available to the adversary in our setting. Second, the underlying gradient-based optimization is computationally demanding. Generating effective patches in real-time is therefore difficult with existing methods, as we demonstrate in Section 5. Although the reliance on camera images could, in principle, be addressed using universal adversarial patches [e.g., 13], optimizing these patches is computationally intensive and therefore infeasible under real-time constraints.

Generative Learning of Patches. As a remedy, we turn to generative learning. Diffusion models, which achieve state-of-the-art performance in image synthesis, can be conditioned on external parameters to generate outputs matching desired properties [1, 10, 15, 55]. This conditioning capability is well suited to our scenario: the model generates patches based on the desired adversarial action and display constraints.

Patch Diffusion Model. We train a diffusion model based on the EDM framework [20] to synthetically generate patches conditioned on parameters of the environment; more specifically, the constraint mask c of the adversarial patch within the final manipulated camera image and a target action $a = (g - s)$ expressed in the drone's frame of reference. We specify the constraint matrix as a transformation matrix applied to the output of the model to map it directly into the constrained region. Furthermore, we tailor the training objective to our setting. Contrary to simply denoising the patches, we additionally incorporate the target detection models into the training. After a denoising step, the patches are projected onto background images randomly drawn from dataset $\mathbb{X}$. The manipulated images are then used as input to the object detector. The output action computed should then be as close as possible to the desired target waypoint defined by the conditioning. Based on Equation (2), the goal is to minimize the expectation of the distance between the predicted and goal action. More specifically, we search for parameters θ of the diffusion model by solving the problem:

$$\underset{\theta}{\operatorname{argmin}} \sum_{x \sim \mathbb{X}, c \sim \mathbb{C}, a \sim \mathbb{A}} \mathcal{L}_\theta(x, c, a) \,, \tag{6}$$

$$\mathcal{L}_\theta(x, c, a) = \|a - \gamma(x \oplus_c p_\theta(a, c))\| \,. \tag{7}$$

Generated patches are therefore guided towards impacting the object detector output depending on conditioning. As shown in Fig. 4, standard adversarial patches and patches generated using a diffusion model for the same action share similar structures, yet continuous versions exhibit a more coarse-grained and smooth appearance. This texture arises because the diffusion model generalizes existing patches and batches of possible background images instead of a single image only.

3.4 Practical Considerations

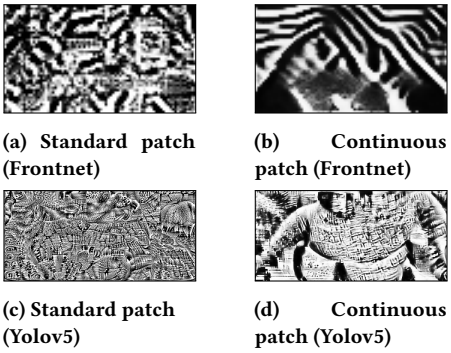
Equipped with a general optimization problem, we consider practical aspects that influence how continuous patches can be generated for real-world object tracking systems.

Detector-Only Tracking Pipelines. Our evaluation targets two object detection architectures with distinct control mappings. PULP-Frontnet [34] is an end-to-end model that directly predicts poses, representative of resource-constrained nano-UAVs. YOLOv5, by contrast, outputs bounding boxes without built-in control logic and is widely adopted in commercial and research drone systems for initial tracking stages [6, 51]. To evaluate YOLOv5, we construct a perception-to-control pipeline using differentiable pose estimation and waypoint generation. Access to complete proprietary pipelines of commercial drones would require time-consuming reverse engineering. Our custom pipeline draws from state-of-the-art methods in drone research [9, 23, 27, 32, 47, 50] and enables evaluation across a broader class of bounding-box-based detectors rather than a single proprietary system. This design choice improves generalizability while maintaining realism for resource-constrained tracking scenarios common in consumer drones.

Differentiable Bounding Boxes. Object detectors map camera images directly to poses or first to bounding boxes (see Section 2). In the latter case, a common post-processing method to efficiently filter and merge redundant and non-relevant boxes is non-maximum suppression. This method neglects all boxes with low likelihoods, and filters redundant boxes by measuring their intersection-over-union. Unfortunately, this step is not differentiable and hinders applying gradient-based optimization. Since we are focusing on predicting the pose of a certain subject, for instance a person, we can filter all boxes of high likelihood for "person" and neglect all other boxes. To approximate this procedure, we first apply the softmax function to map all predicted likelihoods to the range $[0, 1]$, where resulting likelihoods close to 1 correspond to boxes containing a person. We then multiply these weights with the predicted boxes, weighing them by the likelihood of being chosen. Finally, we sum all remaining boxes to merge them. This is viable for merging boxes if we assume the goal of the person detection task is to follow the object inside the camera images that causes the object detector to believe that it is most likely a human. As a result, we obtain an approximate variant of the non-maximum suppression algorithm, enabling gradient flow during diffusion model training.

Differentiable Pose Estimation. 3D pose estimation from 2D images requires calibrated camera intrinsic and extrinsic matrices—intrinsics map pixels to the camera frame, extrinsics map to an external frame (e.g., vehicle). Following Vrba and Saska [47], we use a YOLO detector to localize objects simplified as spheres of known radius, reducing hyperparameters to just radius and camera matrices. This approach is fast and lightweight for real-time use. We construct rays r_1 and r_2 from the optical center to the bounding box edges, plus a center ray $r_c = \frac{1}{2}(r_1 + r_2)$. After normalization, we compute the angle between rays:

$$\alpha = \arccos\left(\frac{r_1 \cdot r_2}{\|r_1\|\|r_2\|}\right) . \tag{8}$$



(a) Standard patch (Frontnet) **(b) Continuous patch (Frontnet)**

(c) Standard patch (Yolov5) **(d) Continuous patch (Yolov5)**

Figure 4: Comparison of adversarial patches for the same action. The continuous patch is created using a diffusion model, the standard patches were generated via gradient descent optimizing the objective defined in Equation (7).

Distance derives from sphere trigonometry: $d = \frac{\sqrt{2}k}{\sin(0.5\alpha)}$, where k is the known radius. Multiplying d by the normalized center ray gives the 3D position in camera frame; applying the inverse extrinsic transforms it to world coordinates. Orientation follows via 2-argument arctangent of the relative position. This method is computationally efficient, requiring only approximate physical size. More sophisticated approaches (e.g., monocular depth networks) add latency without significantly affecting our threat model.

4 Experimental Setup

To evaluate the efficacy of our continuous adversarial patches, the setup must provide a reproducible and realistic benchmark while ensuring safety throughout all experiments. As direct testing of autonomous vehicles in real-world settings would raise safety concerns and introduce errors from uncontrolled factors, we conduct our experiments in a simulated environment. In the following, we introduce this environment, the target models, and training procedures.

Scope and Limitations. We emphasize that this work is a proof-of-concept study. All experiments are conducted in a custom simulation environment, and the threat model assumes knowledge of the victim's detection model and estimated real-time pose. Our goal is to demonstrate the feasibility and upper-bound severity of continuous adversarial patches, not to claim immediate real-world deployability. We release our simulation environment and code to enable further investigation and benchmarking by the community. Full limitations are discussed in Section 7.

4.1 Simulation Environment

Existing simulators offer physically accurate drone simulation with detailed rendering [11], but none specifically benchmark drone robustness against adversarial inputs. Rendering engines introduce unnecessary computational overhead for adversarial testing. Precomputed image datasets suffice for evaluating attack effectiveness.

We develop a custom simulation environment tailored to our experimental needs. Released with our codebase, it allows drone manufacturers to benchmark systems using our setup or adapt it to

their requirements. All components (controller, monitors, camera images, adversarial methods, object detectors) are easily replaceable.

Simulation Approach. The vehicle operates within a confined 3D space (see Fig. 2). A proportional controller [49] computes updated poses given waypoints and pose update frequency. The controller uses position gain $K_p^{\text{pos}} = 1.5$ with velocity clamping at 15 m/s maximum, and yaw rate gain $K_p^{\text{yaw}} = 3.0$ with yaw rate clamping at 3 rad/s maximum. These parameters estimate the specifications provided for the DJI Neo[1]. Camera images are drawn from the extended Frontnet dataset of indoor scenes (see Section 4.2) and processed by an object detection model γ to predict human subject poses. These relative poses map to target waypoints, maintaining a safe distance while keeping the subject centered in the camera's field of view.

Before capturing each image, we simulate an adversary displaying an adversarial patch on a stationary video display of user-defined size. This external manipulation requires no knowledge of the camera image—only the poses of the drone and display. Patches are projected within each camera image via perspective transformations [42].

Simulation Procedure. Each run begins with the autonomous vehicle at a fixed pose s_0 in world coordinates, with the monitor positioned two meters in front and facing the drone. Users specify monitor size via diagonal inches. The simulation executes at 10 Hz, following waypoints derived from the object detection model and translated by the proportional controller. Each configuration is repeated across three random seeds to account for stochastic elements in the diffusion model sampling. Results report averages across runs, with standard deviations indicated where applicable.

4.2 Target Models and Datasets

We evaluate two resource-efficient object detection models commonly deployed on drones: PULP-Frontnet [34] and YOLOv5 [19]. Both models are resource-efficient and perform well at detecting human subjects, making them realistic targets for our adversarial benchmarks [6].

PULP-Frontnet. PULP-Frontnet [34] is an end-to-end pose estimation model designed for the Crazyflie 2.1 nano aerial vehicle. It processes 160×96 grayscale images to directly predict a human subject's 3D position and orientation angle in the drone's frame. These predictions drive the vehicle's controller to maintain the subject in the camera's center, meaning the model's output directly dictates the drone's behavior.

YOLOv5. We utilize YOLOv5 [19], a standard in the YOLO family [37]. It accepts 640×320 RGB images, outputting 12,600 bounding boxes with associated objectness scores and 80 class probabilities per box. Since YOLOv5 lacks end-to-end control capabilities, we construct a perception-to-control pipeline using the differentiable pose estimation and waypoint generation described in Section 3.4. For consistency, we convert inputs to grayscale.

[1]DJI Neo Specifications

PULP-Frontnet Dataset. Our simulation uses an extended version of the PULP-Frontnet dataset [34], originally designed for training end-to-end tracking models. It contains indoor scenes with multiple human subjects, labeled with ground-truth 3D positions and orientation angles. While the original data includes annotations, the extension by Hanfeld et al. [13] adds unannotated images to increase scene variety and flight dynamics. To correct a bias toward subjects appearing on the right side of the frame, we vertically mirror all images, effectively doubling the dataset size.

Dataset Limitations. The Frontnet dataset consists of indoor scenes with controlled lighting and predominantly static backgrounds. While the extension by Hanfeld et al. [13] and our vertical mirroring increase scene variety, the dataset does not include outdoor environments, adverse weather conditions, varying illumination (daylight to darkness), occluded subjects, or multiple simultaneous tracked objects. This limits generalizability but enables controlled evaluation of the attack mechanism.

4.3 Training Procedure

We train the diffusion model of our attack on a dataset of 1000 optimal patches generated using projected gradient descent (PGD) for a diverse set of environmental conditions. More specifically, we sample uniformly random actions and constraints and then generate universal patches similar to [13]. Concretely, for each randomly chosen $c \in \mathbb{C}$ and $a \in \mathbb{A}$, we compute patches $\hat{p} \in \mathbb{P}$ by solving

$$\underset{\hat{p}}{\operatorname{argmin}} \sum_{x \in \mathbb{X}} \left\| a - \gamma(x \oplus_c \hat{p}) \right\|, \tag{9}$$

using gradient descent. This optimization objective mirrors the optimization objective of continuous adversarial patches, differing in that a specific single patch $\hat{p}$ for a chosen constraint and action is required instead of a general patch generating function p.

For our implementation, we follow [20] and train a UNet model [40] with residual blocks [14] as the underlying architecture. To condition the model on a user-specified target, we embed the target using a linear layer followed by a convolutional layer, and add the resulting features as additional channels to the input image. The model is trained for 1000 epochs with a batch size of 32, the Adam optimizer [21] with a learning rate of 10^{-4}, and 1000 maximum denoising time steps. We use Equation (7) as the loss function to be minimized. The training of each diffusion model takes less than four hours utilizing an NVIDIA A100 GPU, making this a feasible attack despite the high number of epochs. Moreover, inference using the trained model requires at most two milliseconds, supporting update rates of up to 500 Hz for the patch generation component alone. We note that end-to-end pipeline latency—including drone pose estimation, patch rendering, and display refresh—would reduce this rate in practice. Note that an optimal patch generation with PGD would be possible at 0.95 ± 0.38 Hz for Frontnet and 0.03 ± 0.05 Hz for Yolov5 in our simulation per patch. Due to batching, the diffusion model could potentially even predict multiple patches at once while still being significantly faster.

5 Evaluation

We examine whether continuous adversarial patches enable sustained control over an autonomous drone across different object detection models in our simulated setting, and how robust this control remains under varying operational parameters. This evaluation is organized around two attack scenarios. We first study *slingshot attacks*, in which the attacker aims to rapidly accelerate the drone along a single target direction. In this setting, we examine factors that influence attack effectiveness in simulation and compare continuous adversarial patches against a range of baseline strategies. We then consider *freestyle attacks*, where the attacker attempts to guide the drone along more complex, multi-step trajectories in simulation.

To ensure robust results, we evaluate several combinations of simulation parameters, repeat each configuration across three random seeds, and report averages across runs. All results are obtained in the custom simulation environment described in Section 4.

5.1 Slingshot Attacks

In the slingshot setting, the attacker specifies a target direction for drone acceleration in simulation. The objective is to induce sustained motion in this direction while keeping the adversarial display within the drone's field of view. We measure success as signed displacement projected onto the target direction—positive values indicate motion in the intended direction.

We evaluate slingshot performance as a function of three factors: target direction, display size, and detection model. Unless stated otherwise, results are averaged across display sizes and directions. Overview results for YOLOv5 are in Fig. 5; Frontnet results appear in Appendix D.

Target Direction. We study four trajectories: forward, backward, left, and right. Attack performance varies substantially—forward motion yields the strongest displacement, followed by backward, while leftward and rightward motion result in smaller displacement. To avoid overstating performance aligned with inherent dataset drift, we focus on the *backward* slingshot direction for subsequent evaluation, as it is least affected by bias and particularly challenging since the visible display area shrinks as the drone moves away.

Display Size. We vary the display diagonal across 60, 80, 100, and 120 inches (16:9 aspect ratio), covering commercial monitors. Attack effectiveness increases with display size, as larger displays allow greater control over the camera image. Even 60-inch displays consistently enable slingshot maneuvers.

Displacement eventually saturates for each size due to the display effectively shrinking as the drone moves away. From this saturation point, we estimate an attacker must control at least 7 % of the camera's field of view to succeed.

Detection Model. While slingshot attacks succeed for both Frontnet and YOLOv5, the attack on Frontnet struggles with continued control. Counterintuitively, despite YOLOv5's stronger benign performance, its higher input resolution provides more degrees of freedom for the attacker when the drone moves backward.

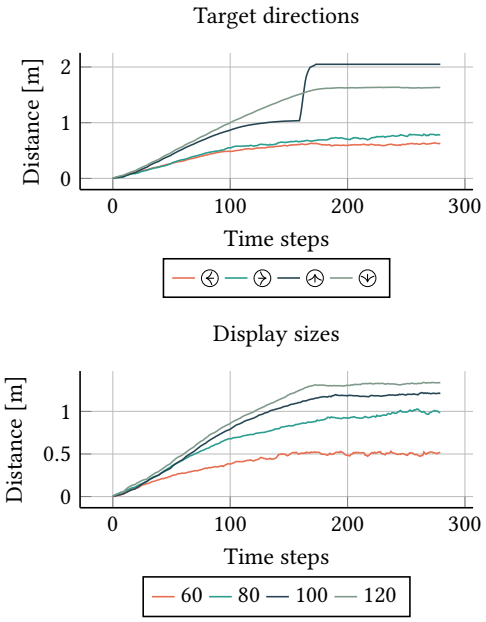


Figure 5: Sensitivity of slingshot attacks to target direction and display size. Results are averaged across all directions except forward for display sizes and all display sizes for target directions.

5.2 Baseline and Alternative Attacks

We compare slingshot attacks against (i) baseline strategies lacking online adaptability, and (ii) alternative patch-generation mechanisms. Performance is normalized by maximum achievable distance in the target direction. Table 1 reports effectiveness for backward slingshot, averaged across display sizes.

Baselines. We consider unaltered camera input (*None*), static adversarial patches optimized for limited placements [13], and an artificial PGD attack constrained to 10 Hz with direct camera access (unrealistic; serves as an upper bound).

Across both models, these baselines fail to induce sustained slingshot motion. Static patches and PGD occasionally cause isolated deviations, but remain near zero, indicating inability to counteract inherent motion tendencies. PGD would require higher computation time (0.95 ± 0.38 Hz for Frontnet, 0.03 ± 0.05 Hz for YOLOv5), leading to different robot states post-optimization.

Alternative Patch Generation. Beyond our diffusion-based approach, we evaluate precomputed patch alternatives:

- *Corpus:* 1000 universal adversarial patches from training; nearest neighbor selected based on desired action and display mask.
- *Interpolation:* Two nearest neighbors retrieved and interpolated.

Both yield modest improvements over static baselines for Frontnet but fail on YOLOv5. Limited by reliance on precomputed patches, they cannot generalize effectively.

Table 1: Performance of different slingshot approaches in percent (simulation results). Quality is measured relative to the optimal flight path and averaged across different display sizes. Negative numbers indicate opposite-direction steering. Higher is better.

	Baselines			*Alternative*		*Generative*
Model	**None**	**FAP [13]**	**PGD**	**Interp.**	**Corpus**	**Ours**
Frontnet	−1.3 ± 0.0	1.7 ± 0.6	3.3 ± 1.9	5.8 ± 2.6	18.8 ± 4.8	32.4 ± 7.0
YOLOv5	−0.4 ± 0.0	5.5 ± 2.4	−6.2 ± 1.6	−2.1 ± 0.4	−1.4 ± 0.5	75.3 ± 6.1

In contrast, our diffusion-based approach substantially outperforms all alternatives, achieving 38.3 % for Frontnet and 74.5 % for YOLOv5 under real-time constraints. Sustained slingshot control requires both temporal continuity and an expressive generative model.

5.3 Freestyle Attacks

We extend beyond single-direction slingshot attacks to complex paths. Specifically, we consider a *triangle* and *infinity symbol* trajectory, each defined by 25 waypoints the drone must follow via continuous patches on a video display.

Performance Measure. Since displacement does not capture path-following, we use Fréchet distance—a standard curve similarity measure accounting for spatial deviation and waypoint ordering.

Results. We evaluate the same methods as the slingshot setting; results appear in Table 2. Our approach shows decent performance on complex maneuvers, requiring both trajectory adherence and keeping the monitor in view.

Our diffusion-based attack performs best on YOLOv5 (lowest Fréchet distance) and comparably on Frontnet. Example trajectories appear in Fig. 6 and Appendix Section D. Though the drone does not perfectly trace the target shape, intended trajectories are clearly discernible, while best baselines fail to follow them meaningfully.

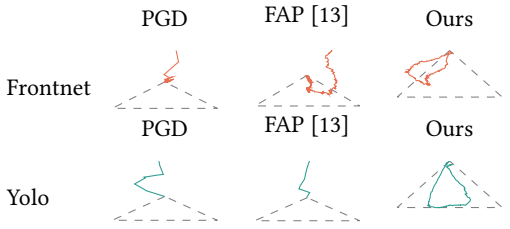


Figure 6: Flight paths of our continuous adversarial patch attack on the triangle trajectory in simulation. The visualized path corresponds to the median run, selected based on Fréchet distance to the target trajectory.

6 Countermeasures

Given the threat of continuous adversarial patches, we outline potential countermeasures that warrant investigation. We organize countermeasures into three categories: detection-based approaches, robustness training, and operational constraints. Evaluating these countermeasures remains future work.

Detection-Based Defenses. Adversarial patches are not a new problem, and many countermeasures have been proposed to address this threat. Two complementary lines of research dominate the field: detecting anomalous inputs and training models to be robust. While detection might appear straightforward, as patches are highly suspicious to human observers, these detectors can be defeated by incorporating them into the adversarial objective and have therefore proven insufficient [4].

Recent work on projection-based attacks has proposed specialized detection mechanisms. DeGhost [33] introduces a CNN classifier with Fourier analysis to distinguish projected objects from real ones, achieving AUC 0.998 on projector-based phantom attacks. Adapting such frequency-domain analysis to video displays (which lack projector artifacts) and testing against our continuous patch attack is promising future work. Furthermore, it is unclear how a vehicle should respond once an attack is detected. A drone could stop following all persons, yet this would introduce a risk of denial-of-service attacks that obstruct the correct functioning of the autonomous vehicle.

Robustness Training. A more fruitful line of research has focused on robustness, particularly through adversarial training [16, 28]. While adversarial training often yields only moderate protection in practice against static patches, it may offer stronger defense against continuous adversarial patches given their adaptive nature. In combination with our generative approach, defenders could potentially train on large amounts of continuously-generated adversarial inputs, creating a dynamic arms race. We therefore argue that adversarial training is a promising avenue for defense, but is likely insufficient as a standalone measure without additional safeguards. Evaluating adversarial training against continuous patches in simulation, then validating in physical tests, is a priority for future work.

Operational and Sensor-Based Defenses. Some high-end drones include obstacle avoidance mechanisms [41], but these are uncommon in consumer models used for personal tracking. Limiting tracking speed could improve safety but trades utility for responsiveness. Since attack success scales with display size (Fig. 5), constraining operational environments to avoid large displays may reduce risk in security-sensitive settings. Multi-modal sensor fusion (e.g., LiDAR, radar) could also help identify the planar structure of a display, though prior work has shown adversarial objects can target LiDAR systems [45]. Testing continuous patches against multi-modal perception remains future work.

Table 2: Path following capability of proposed freestyle maneuvers (simulation results). Performance is average Fréchet distance from actual to target trajectory. Lower is better.

	Baselines			*Alternative*		*Generative*
Model	**None**	**FAP [13]**	**PGD**	**Interp.**	**Corpus**	**Ours**
Frontnet	1.35 ± 0.15	1.07 ± 0.02	1.36 ± 0.07	1.53 ± 0.03	1.09 ± 0.03	1.11 ± 0.04
YOLOv5	1.14 ± 0.02	1.55 ± 0.04	1.62 ± 0.04	1.50 ± 0.03	1.44 ± 0.03	0.52 ± 0.01

7 Limitations

While we have tried to ensure that both our attack and simulation are as realistic as possible, significant limitations remain. We organize these into categories reflecting the major concerns raised during peer review.

Simulation-Only Evaluation. This is the primary limitation of our work. We evaluate our attack solely in a simulated environment. While this choice is justified by reproducibility, safety, and computational efficiency, it means our results do not demonstrate physical realizability. Critical factors not modeled include: (i) camera artifacts (rolling shutter, exposure adjustment, motion blur, lens distortion), (ii) display artifacts (refresh rate limitations, color gamut mismatch, viewing angle distortion, brightness constraints), (iii) environmental conditions (ambient lighting variations from daylight to darkness, weather, wind gusts affecting drone stability), (iv) full drone dynamics (propeller thrust limits, battery degradation, payload effects, motor response latency), and (v) robustness modules (Kalman filtering, sensor fusion, re-identification smoothing). These omissions may cause our simulation to overestimate attack success rates. Physical validation on real drones is necessary and remains future work.

Custom YOLOv5 Pipeline. For our YOLOv5 evaluation, we constructed a perception-to-control pipeline using differentiable pose estimation and waypoint generation. This pipeline is our own design. While YOLOv5 is widely used in drone research [6, 51], commercial implementations typically include additional components (ReID networks, trajectory smoothing, obstacle avoidance integration) that may alter attack effectiveness. Our results therefore establish an upper bound on vulnerability for bounding-box-based detectors, but actual deployed systems may be more or less susceptible depending on their complete architecture. However, the authors of FlyTrap [53] were able to successfully transfer their white-box attacks to black-box consumer drones.

Controller Simplifications. Our simulation uses a proportional controller without re-identification components or temporal smoothing of the tracked object's position. Many consumer drones incorporate these features to handle occlusions and maintain tracking stability. Omitting them simplifies the simulation and aligns with prior adversarial drone research [13], but may overstate attack effectiveness compared to production systems with more sophisticated tracking stacks. We leave ablation studies on controller complexity and ReID integration for future work.

White-Box Assumption. We assume the attacker possesses white-box access to the victim's detection model, camera model, and calibration parameters. This strong assumption enables worst-case vulnerability probing. While transfer attacks against surrogate models are feasible [52] and pose estimation from RGB imagery is achievable [22, 42], real-world attackers would face uncertainty about system internals. Gray-box and black-box attack effectiveness against continuous adversarial patches remains unquantified.

Training Data Diversity. The diffusion model is trained on 1000 PGD-generated patches covering uniform samples of action space and constraint masks from the Frontnet dataset. However, this dataset consists of indoor scenes with controlled lighting and limited object occlusion scenarios. It does not include outdoor environments, adverse weather conditions, varying illumination (daylight to darkness), multiple simultaneous tracked objects, or unusual camera angles. The attack's generalization to out-of-distribution conditions is therefore unknown and warrants investigation.

Fixed-Size Patch Resolution. We generate patches at fixed resolution and apply perspective transformations to map them into the camera image. In practice, the apparent patch size varies with drone distance to the display, and very small patches (at long distances) may lose adversarial effectiveness due to pixelation or insufficient coverage. Our simulation does not explicitly model this degradation at extreme ranges.

Pose Estimation Requirements. Our attack assumes the adversary can estimate the drone's pose in real time with sufficient accuracy. While pose estimation from stereo cameras or RGB imagery is feasible [18, 22, 36], estimation errors accumulate over time and may reduce attack effectiveness. We do not quantify the attack's tolerance to pose estimation noise or latency.

Defense Evaluation Gap. We do not evaluate any defensive countermeasures empirically. While we outline potential defenses in Section 6, systematic testing of adversarial training, detection-based approaches, or multi-modal fusion against continuous adversarial patches is left for future work. This gap is particularly important given that defenses evaluated against static patches may not generalize to adaptive attacks.

8 Related Work

Our work connects research on adversarial robustness in autonomous systems, physical adversarial patches, and diffusion-based attack generation.

Attacks on Autonomous Vehicles. Early studies demonstrated that visual perturbations can misclassify traffic signs or critical objects, compromising control decisions [3, 12, 45, 46, 54]. Chahe et al. [5] display adversarial patches on a monitor mounted to a ground robot

to force traffic sign misclassification, but require the attacker to navigate to a pre-optimized position and rely on hardware-specific augmentations.

Attacks on Drones. Tian et al. [44] perturb steering angles via single-image attacks, but these lack trajectory control. Hanfeld et al. [13] introduce "Flying Adversarial Patches"—static patches on an attacker drone to kidnap a victim drone. However, as reproduced in our results (Table 1, Table 2), these static patches fail to adapt to environmental changes, and deployment is hindered by aerodynamic effects from large physical patches.

FlyTrap [53] performs static "distance pulling" via a patch on an umbrella, forcing a drone to collide with the holder. While it avoids pose estimation by operating in pixel space, it fails if the umbrella holder is not the primary tracked target and cannot adapt to dynamic scenes.

DeGhost [33] studies phantom attacks where projectors cast objects onto surfaces, causing drones to misidentify projections as real. They conduct real-world experiments on DJI Mavic Air and Tello with YOLOv3, and propose a CNN-based defense. However, DeGhost uses natural images rather than adversarially optimized patterns, relies on static or pre-recorded projections, and drone trajectories emerge from the tracking algorithm rather than being attacker-specified. Our work differs in three respects: (i) we use video displays rather than projectors, (ii) our patches are adversarially optimized and state-conditioned, and (iii) our attack enables attacker-defined trajectories.

Diffusion Models for Adversarial Examples. Chen et al. [7] couple PGD with diffusion guidance for natural adversarial examples. Lin et al. [26] and Wang et al. [48] generate stealthy patches for object vanishing, printable on clothing. Miao et al. [29] guide generation toward adversarial subspaces of semantic groups. Li et al. [25] generate car-obscuring patches conditioned on precomputed masks for autonomous driving. Despite their visual stealth, these methods focus on static image generation and do not address the dynamic, temporal constraints of real-time robotic control.

9 Conclusion

Our proof-of-concept study suggests the adversarial attack surface on autonomous vehicles may be broader than previously recognized. Using a diffusion model to generate continuous adversarial patches in real time, we demonstrate in simulation that attackers can not only cause mispredictions but also steer vehicle trajectories along defined paths–a qualitative shift from isolated misclassifications to sustained manipulation.

This contributes a new facet to adversarial machine learning. While prior work has extensively studied adversarial patches for detection, tracking, and navigation, these focused on instantaneous errors. By extending the threat model to include video displays, we show adversarial patches can be dynamic, enabling attacks that unfold over time.

Our findings underscore the need for security measures in autonomous vehicles, though physical validation remains necessary before real-world risk can be substantiated. Manufacturers and researchers should investigate how learning models integrate into vehicle decision-making, as systems must withstand coordinated manipulations—a notable challenge given limited defenses. We release our simulation environment and code to facilitate further investigation, encouraging future work on defense evaluation, gray-box transferability, and physical realizability.

Acknowledgments

This work was funded by the German Federal Ministry of Research, Technology and Space (BMFTR) under the grant "AIgenCY" (16KIS2012), the European Research Council (ERC) under the consolidator grant "MALFOY" (101043410), and Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under project "ALISON" (492020528).

References

[1] Christopher M Bishop and Hugh Bishop. 2023. *Deep learning: Foundations and concepts.* Springer Nature.

[2] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. 2017. Adversarial Patch. *Computing Research Repository (CoRR)* (2017).

[3] Yulong Cao, Chaowei Xiao, Anima Anandkumar, Danfei Xu, and Marco Pavone. 2022. AdvDO: Realistic Adversarial Attacks for Trajectory Prediction. In *Computer Vision – ECCV 2022.* Springer Nature Switzerland, 36–52.

[4] Nicholas Carlini and David A. Wagner. 2017. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. In *Proc. of the ACM Workshop on Artificial Intelligence and Security (AISec).*

[5] Amirhosein Chahe, Chenan Wang, Abhishek Jeyapratap, Kaidi Xu, and Lifeng Zhou. 2024. Dynamic Adversarial Attacks on Autonomous Driving Systems. In *Robotics: Science and Systems (RSS).*

[6] Chunling Chen, Ziyue Zheng, Tongyu Xu, Shuang Guo, Shuai Feng, Weixiang Yao, and Yubin Lan. 2023. YOLO-Based UAV Technology: A Review of the Research and Its Applications. *Drones* 7, 3 (2023). doi:10.3390/drones7030190

[7] Xinquan Chen, Xitong Gao, Juanjuan Zhao, Kejiang Ye, and Cheng-Zhong Xu. 2023. AdvDiffuser: Natural Adversarial Example Synthesis with Diffusion Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).*

[8] Jun Cheng, Penglei Liu, Qieshi Zhang, Hui Ma, Fei Wang, and Jin Zhang. 2021. Real-Time and Efficient 6-D Pose Estimation From a Single RGB Image. *IEEE Transactions on Instrumentation and Measurement* 70 (2021), 1–14. doi:10.1109/TIM.2021.3115564

[9] Yutao Cui, Tianhui Song, Gangshan Wu, and Limin Wang. 2023. Mixformerv2: Efficient fully transformer tracking. *Advances in neural information processing systems* 36 (2023), 58736–58751.

[10] Prafulla Dhariwal and Alexander Nichol. 2021. Diffusion Models Beat GANs on Image Synthesis. In *Advances in Neural Information Processing Systems*, Vol. 34. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2021/file/49ad23d1ec9fa4bd8d77d02681df5cfa-Paper.pdf

[11] Cora A. Dimmig, Giuseppe Silano, Kimberly McGuire, Chiara Gabellieri, Wolfgang Hönig, Joseph Moore, and Marin Kobilarov. 2025. Survey of Simulators for Aerial Robots: An Overview and In-Depth Systematic Comparisons [Survey]. *IEEE Robotics & Automation Magazine* 32, 2 (2025), 153–166. doi:10.1109/MRA.2024.3433171

[12] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust physical-world attacks on deep learning visual classification. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 1625–1634.

[13] Pia Hanfeld, Khaled Wahba, Marina M.-C. Höhne, Michael Bussmann, and Wolfgang Hönig. 2023. Kidnapping Deep Learning-based Multirotors using Optimized Flying Adversarial Patches. In *International Symposium on Multi-Robot and Multi-Agent Systems (MRS).* doi:10.1109/MRS60187.2023.10416782

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* doi:10.1109/CVPR.2016.90

[15] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf

[16] Ashish Hooda, Neal Mangaokar, Ryan Feng, Kassem Fawaz, Somesh Jha, and Atul Prakash. 2023. Theoretically Principled Trade-off for Stateful Defenses against Query-Based Black-Box Attacks. *Computing Research Repository (CoRR)* (2023).

[17] Thomas Georg Jantos, Mohamed Amin Hamdad, Wolfgang Granig, Stephan Weiss, and Jan Steinbrener. 2023. PoET: Pose estimation transformer for single-view, multi-object 6D pose estimation. In *Conference on robot learning.* PMLR, 1060–1070.

[18] Ren Jin, Jiaqi Jiang, Yuhua Qi, Defu Lin, and Tao Song. 2019. Drone Detection and Pose Estimation Using Relational Graph Networks. *Sensors* 19, 6 (2019). doi:10.3390/s19061479

[19] Glenn Jocher. 2020. *YOLOv5 by Ultralytics*. doi:10.5281/zenodo.3908559

[20] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. 2022. Elucidating the Design Space of Diffusion-Based Generative Models. In *Advances in Neural Information Processing Systems*.

[21] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*.

[22] Yo-Chung Lau, Kuan-Wei Tseng, Peng-Yuan Kao, I-Ju Hsieh, Hsiao-Ching Tseng, and Yi-Ping Hung. 2023. Real-Time Object Pose Tracking System With Low Computational Cost for Mobile Devices. *IEEE Journal of Indoor and Seamless Positioning and Navigation* 1 (2023), 211–220. doi:10.1109/JISPIN.2023.3340987

[23] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. 2019. SiamRPN++: Evolution of Siamese Visual Tracking With Very Deep Networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4277–4286. doi:10.1109/CVPR.2019.00441

[24] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. 2018. High performance visual tracking with siamese region proposal network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8971–8980.

[25] Jing Li, Zigan Wang, and Jinliang Li. 2024. AdvDenoise: Fast Generation Framework of Universal and Robust Adversarial Patches Using Denoise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[26] Shuo-Yen Lin, Ernie Chu, Po-Hung Yeh, Jun-Cheng Chen, and Jia-Ching Wang. 2025. Diffusion to Confusion: Naturalistic Adversarial Patch Generation Based on Diffusion Model for Object Detector. In *2025 IEEE International Conference on Image Processing (ICIP)*. doi:10.1109/ICIP55913.2025.11084754

[27] Juanqin Liu, Leonardo Plotegher, Eloy Roura, Cristino de Souza Junior, and Shaoming He. 2025. Real-Time Detection for Small UAVs: Combining YOLO and Multiframe Motion Analysis. *IEEE Trans. Aerospace Electron. Systems* 61, 5 (2025), 13419–13433. doi:10.1109/TAES.2025.3577592

[28] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *Proc. of the International Conference on Learning Representations (ICLR)*.

[29] Boming Miao, Chunxiao Li, Yao Zhu, Weixiang Sun, Zizhe Wang, Xiaoyi Wang, and Chuanlong Xie. 2024. AdvLogo: Adversarial Patch Attack against Object Detectors based on Diffusion Models. *CoRR* abs/2409.07002 (2024). https://doi.org/10.48550/arXiv.2409.07002

[30] Stefan Mitsch, Khalil Ghorbal, David Vogelbacher, and André Platzer. 2017. Formal verification of obstacle avoidance and navigation of ground robots. *Int. J. Robotics Res.* 36, 12 (2017).

[31] Seyed Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 86–94. doi:10.1109/CVPR.2017.17

[32] Paraskevi Nousi, Ioannis Mademlis, Iason Karakostas, Anastasios Tefas, and Ioannis Pitas. 2019. Embedded UAV Real-Time Visual Object Detection and Tracking. In *2019 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. 708–713. doi:10.1109/RCAR47638.2019.9043931

[33] Hotaka Oyama, Ryo Iijima, and Tatsuya Mori. 2024. DeGhost: Unmasking Phantom Intrusions in Autonomous Recognition Systems. In *2024 IEEE 9th European Symposium on Security and Privacy (EuroS&P)*. 78–94. doi:10.1109/EuroSP60621.2024.00013

[34] Daniele Palossi, Nicky Zimmerman, Alessio Burrello, Francesco Conti, Hanna Muller, Luca Maria Gambardella, Luca Benini, Alessandro Giusti, and Jerome Guzzi. 2022. Fully Onboard AI-Powered Human-Drone Pose Estimation on Ultralow-Power Autonomous Flying Nano-UAVs. *IEEE Internet of Things Journal* 9, 3 (2022), 1913–1929. doi:10.1109/JIOT.2021.3091643

[35] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2017. Practical Black-Box Attacks against Machine Learning. In *Proc. of the ACM Asia Conference on Computer and Communications (AsiaCCS)*.

[36] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. 2019. PVNet: Pixel-Wise Voting Network for 6DoF Pose Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[37] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Cnference on Computer Vision and Pattern Recognition*. 779–788.

[38] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: better, faster, stronger. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 7263–7271.

[39] R. Risack, N. Mohler, and W. Enkelmann. 2000. A video-based lane keeping assistant. In *Proc. of the IEEE Intelligent Vehicles Symposium*.

[40] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer International Publishing, Cham.

[41] Shenzhen Da-Jiang Innovations Sciences and Technologies (DJI). 2026. *Introduction to the Aircraft Obstacle Avoidance System*.

[42] Richard Szeliski. 2022. *Computer vision: algorithms and applications*. Springer Nature.

[43] Simen Thys, Wiebe Van Ranst, and Toon Goedeme. 2019. Fooling Automated Surveillance Cameras: Adversarial Patches to Attack Person Detection. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 49–55. doi:10.1109/CVPRW.2019.00012

[44] Jiwei Tian, Buhong Wang, Rongxiao Guo, Zhen Wang, Kunrui Cao, and Xiaodong Wang. 2022. Adversarial Attacks and Defenses for Deep-Learning-Based Unmanned Aerial Vehicles. *IEEE Internet of Things Journal* 9, 22 (2022), 22399–22409. doi:10.1109/JIOT.2021.3111024

[45] James Tu, Mengye Ren, Siva Manivasagam, Ming Liang, Bin Yang, Richard Du, Frank Cheng, and Raquel Urtasun. 2020. Physically Realizable Adversarial Examples for LiDAR Object Detection. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[46] Meriel von Stein, David Shriver, and Sebastian Elbaum. 2023. DeepManeuver: Adversarial Test Generation for Trajectory Manipulation of Autonomous Vehicles. *IEEE Transactions on Software Engineering* 49, 10 (2023), 4496–4509. doi:10.1109/TSE.2023.3301443

[47] Matouš Vrba and Martin Saska. 2020. Marker-Less Micro Aerial Vehicle Detection and Localization Using Convolutional Neural Networks. *IEEE Robotics and Automation Letters* 5, 2 (2020), 2459–2466. doi:10.1109/LRA.2020.2972819

[48] Zhixiang Wang, Xingjun Ma, and Yu-Gang Jiang. 2025. BadPatch: Diffusion-Based Generation of Physical Adversarial Patches. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*.

[49] Mark J Willis. 1999. Proportional-integral-derivative control. *Dept. of Chemical and Process Engineering University of Newcastle* 6 (1999).

[50] Hsiang-Huang Wu, Zejian Zhou, Ming Feng, Yuzhong Yan, Hao Xu, and Lijun Qian. 2019. Real-Time Single Object Detection on The UAV. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. 1013–1022. doi:10.1109/ICUAS.2019.8797866

[51] Xin Wu, Wei Li, Danfeng Hong, Ran Tao, and Qian Du. 2022. Deep Learning for Unmanned Aerial Vehicle-Based Object Detection and Tracking: A survey. *IEEE Geoscience and Remote Sensing Magazine* 10, 1 (2022), 91–124. doi:10.1109/MGRS.2021.3115137

[52] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L. Yuille. 2019. Improving Transferability of Adversarial Examples With Input Diversity. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[53] Shaoyuan Xie, Mohamad Habib Fakih, Junchi Lu, Fayzah Alshammari, Ningfei Wang, Takami Sato, Halima Bouzidi, Mohammad Abdullah Al Faruque, and Qi Alfred Chen. 2026. FlyTrap: Physical Distance-Pulling Attack Towards Camera-based Autonomous Target Tracking Systems. In *ISOC Network and Distributed System Security Symposium (NDSS)*.

[54] Xing Xu, Jingran Zhang, Yujie Li, Yichuan Wang, Yang Yang, and Heng Tao Shen. 2021. Adversarial Attack against Urban Scene Segmentation for Autonomous Vehicles. *IEEE Transactions on Industrial Informatics* 17, 6 (2021), 4117–4126. doi:10.1109/TII.2020.3024643

[55] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023. Adding Conditional Control to Text-to-Image Diffusion Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

A Use of Generative AI

Generative AI has been used as part of the attack method in the form of a self-developed and trained diffusion model generating adversarial patches. We used an LLM (*Gemini 3.1 Pro*) only to correct and improve grammar and writing style of individual paragraphs and generate Fig. 3. We did not let the LLM generate parts of the paper without major revision by the first author. We furthermore utilized *Qwen 3 Coder Next* for in-line code completion and refactoring.

B Ethical Considerations

This work exposes a new attack surface in autonomous vehicles, particularly relevant for consumer drones that track individuals. We discuss potential defense strategies (Section 6) and release our simulation environment to enable manufacturer auditing, arguing that these benefits outweigh the risks of revealing a new attack vector. As our assumptions about real-world deployment are hypothetical, we have not notified specific vendors.

All experiments are conducted in simulation with no human subject testing. The only personal data used consists of previously published images from the PULP-Frontnet dataset, so no uninvolved individuals' privacy is affected.

C Open Science

To foster future research and reproducibility of our findings, we make our simulation environment, attack implementations, and experimental scripts publicly available.

github.com/mlsec-group/continuous-patches

For ethical reasons, however, we do not release the trained diffusion model weights. Although these models would facilitate reproduction of our experimental evaluation, they could also be readily misused by adversaries to generate continuous adversarial patches in practice. To balance reproducibility with ethical considerations, we instead provide scripts that allow the diffusion models to be trained given sufficient hardware resources and technical expertise. We consider this a reasonable trade-off between open scientific practice and the mitigation of potential misuse.

D Additional results

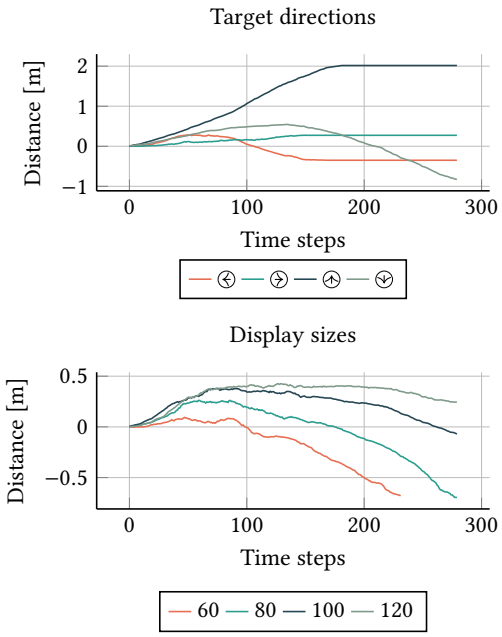


Figure 7: Sensitivity of slingshot attacks to target direction and display size. Results are averaged across all directions except forward for display sizes and all display sizes for the target directions.

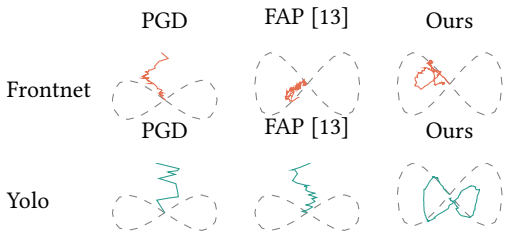


Figure 8: Flight paths of our continuous adversarial patch attack on an infinity sign. The visualized path corresponds to the median run, selected based on its Fréchet distance to the target trajectory.