VOGUES: Validation of Object Guise using Estimated Components

Raymond Muller, *Purdue University;* Yanmao Man and Ming Li, *University of Arizona;* Ryan Gerdes, *Virginia Tech;* Jonathan Petit, *Qualcomm;* Z. Berkay Celik, *Purdue University*

https://www.usenix.org/conference/usenixsecurity24/presentation/muller

This paper is included in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

Open access to the Proceedings of the 33rd USENIX Security Symposium is sponsored by USENIX.

VOGUES: Validation of Object Guise using Estimated Components

Raymond Muller[1], Yanmao Man[2], Ming Li[2], Ryan Gerdes[3], Jonathan Petit[4], and Z. Berkay Celik[1]

[1]Purdue University, *{mullerr, zcelik}@purdue.edu*
[2]University of Arizona, *{yman, lim}@arizona.edu*
[3]Virginia Tech, *rgerdes@vt.edu*
[4]Qualcomm, *petit@qti.qualcomm.com*

Abstract

Object Detection (OD) and Object Tracking (OT) are an important part of autonomous systems (AS), enabling them to perceive and reason about their surroundings. While both OD and OT have been successfully attacked, defenses only exist for OD. In this paper, we introduce VOGUES, which combines perception algorithms in AS with logical reasoning about object components to model human perception. VOGUES leverages pose estimation algorithms to reconstruct the constituent components of objects within a scene, which are then mapped via bipartite matching against OD/OT predictions to detect OT attacks. VOGUES's component reconstruction process is designed such that attacks against OD/OT will not implicitly affect its performance. To prevent adaptive attackers from simultaneously evading OD/OT and component reconstruction, VOGUES integrates an LSTM validator to ensure that the component behavior of objects remains consistent over time. Evaluations in both the physical domain and digital domain yield an average attack detection rate of 96.78% and an FPR of 3.29%. Meanwhile, adaptive attacks against VOGUES require perturbations 30× stronger than previously established in OT attack works, significantly increasing the attack difficulty and reducing their practicality.

1 Introduction

Autonomous systems (AS) with camera-focused perception pipelines are employed in various domains, including autonomous vehicles [15, 31], automated surveillance [34], and drones [40]. Central to these pipelines is object detection (OD), which identifies the class and size of surrounding objects to give the system an understanding of the environment. The importance of OD has made it a target of various attacks, including object deletion [22], which can hide safety-critical objects from a system; object misclassification [10], which mislabels certain objects in order to influence autonomous planning; and object creation [37], which causes the target system to hallucinate objects that do not exist. To combat these attacks, OD defenses have leveraged methods such as feature processing [56], which reduces the available search space needed to conduct an attack, certified robustness [53], which aims to formally guarantee a certain degree of resistance against adversaries, and contextual consistency [16], which leverages context information such as the movement and position of objects to detect attacks.

Despite their efficacy, these defenses have limitations. Spatial smoothing can be circumvented by attackers aware of its usage [5], certified robustness faces scalability challenges due to complex mathematical processes, and contextual consistency suffers in environments lacking contextual cues (e.g., for or darkness). To increase the robustness of perception, object tracking (OT) has been added to modern AS perception pipelines, working on top of OD to associate the same objects over time across multiple video frames. OT enforces *spatiotemporal consistency* onto the OD results, ensuring that there are no impossible changes in the velocity or trajectories of tracked objects. This boosts tracking accuracy and resilience against OD attacks [24].

Recent work has shown that OT is vulnerable to a variety of attacks, once again putting the perception pipeline at risk. The attacks include *tracker-hijacking* [24, 36], where an attacker moves the predicted bounding box of a tracked object to cause physical consequences such as vehicle collisions, and *cooling-shrinking* [57], where an attacker disables OT entirely to hide safety-critical objects.

Because OD and OT have different objectives, OD defenses do not work against OT attacks. For example, a recent work that leverages contextual consistency against object misclassification [33] examines object movements in OD to verify the correctness of perception results. Yet, it does not consider whether the movement itself is modified by an attacker to behave in a consistent manner, e.g., via tracker hijacking. Such an attack would be outside the scope of OD and its attacks. Although a single defense against OT attacks has been proposed [23], it is not practical for real-world use. The defense creates perturbation-canceling noise to counter the effects of tracker hijacking. Yet, when no attack is conducted, this noise

can unintentionally hijack its own tracking results [36].

In this paper, we introduce VOGUES, the first work to detect tracker hijacking and cooling-shrinking attacks against OT. Inspired by neuro-symbolic AI [47] and sensing-reasoning [58], we combine distinct neural networks to model human visual reasoning, which verifies that the constituent components (e.g., the bumpers/roof/wheels of vehicles, the head/limbs/torso for pedestrians) of a seen object are detectable alongside the object itself. We first use a Spatial Transform Network (STN) and Region Proposal Networks (RPN) to isolate objects within an image frame into tightly cropped instances. Next, we extend *pose estimation* models, used by modern AS such as Waymo [50], to extract *key points* we define as important components for objects of a specified class. Lastly, we process the key points into *component sets*, one per object in the image frame, and leverage bipartite matching to check that these component sets are consistent with our OD and OT results. If an object's component set is separated from its tracker, this indicates an attack.

When extracting component sets, we select algorithms with significantly different weaknesses than OD and OT algorithms. This minimizes the likelihood of OD/OT attacks from implicitly affecting our defense. To further guard against attacks simultaneously targeting both our component extraction algorithms and OD/OT pipeline, we train an LSTM verifier. The verifier fuses class-based spatiotemporal consistency checking with our component extraction algorithms to detect tampering in component extraction.

We evaluate the efficacy of VOGUES for both physical and digital-domain attacks against object tracking and detection. We consider non-adaptive attackers, who use the same threat model as previous work and are unaware of VOGUES, and adaptive attackers, who are aware of VOGUES and actively alter their attacks to evade it. For instance, to conduct a successful tracker hijacking attack, an attacker would need to simultaneously hijack the object tracker and spoof the object's component reconstruction results to be spatiotemporally consistent over time with respect to its class.

VOGUES achieves a detection rate of 99.49% for non-adaptive attacks in the digital domain and 93.88% for non-adaptive attacks in the physical domain. In benign settings, VOGUES has a false positive rate of 1.78% in the AV and 2.98% in the autonomous surveillance domains. Additionally, for an adaptive attack to simultaneously compromise perception and evade VOGUES, the perturbations must be 49.9% stronger than a non-adaptive attack. Thus, adaptive attacks launched in the digital domain are unsuccessful against VOGUES when launched in the physical domain, requiring a more powerful perturbation method than proposed in prior work [21, 36].

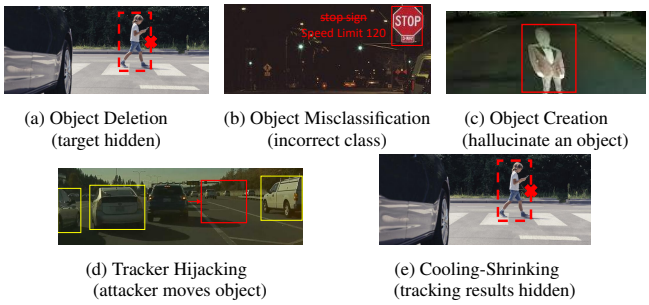
In this paper, we make the following contributions:

- We introduce VOGUES[1], the first practical detection framework for tracker hijacking and cooling-shrinking attacks against object tracking. VOGUES includes systematic approaches to isolate and extract object components via a neural network pipeline and check them against OD/OT results through bipartite matching.

- To improve robustness and increase the difficulty of adaptive attacks, we propose a special LSTM verifier to detect invalid sequences of extracted components over time.

- VOGUES achieves an average detection rate of 99.49% against attacks in the digital domain and 94.06% against real-world attacks, with an overall false positive rate of 3.29%. We also demonstrate the difficulty of employing adaptive attacks against VOGUES in the real world.

[1] Available at https://github.com/purseclab/VOGUES



(a) Object Deletion (target hidden)
(b) Object Misclassification (incorrect class)
(c) Object Creation (hallucinate an object)
(d) Tracker Hijacking (attacker moves object)
(e) Cooling-Shrinking (tracking results hidden)

Figure 1: Attacks against OD (top) and OT (bottom).

2 Background and Related Work

2.1 Object Detection and Tracking

Autonomous systems (AS) used in different domains have similar perception pipelines. The first component of their perception pipelines is classifying the surrounding objects and determining their sizes. This is the task of object detection (OD). Object detectors in AS are typically built on Convolutional Neural Networks (CNNs) designed to yield results in real time. YOLO [44] is a popular architecture used in many domains, including AS. It has high speed and maintains competitive performance compared to other popular architectures such as Faster R-CNN [45].

The next stage in the perception pipeline is object tracking (OT), which aims to associate the same objects over time across frames. There are two main methods. First, tracking-by-detection (e.g., SORT [4]), predicts bounding boxes using a Kalman Filter and associates objects based on similarity metrics (e.g., Intersection over Union). Second, the more recent Siamese trackers [41] employ Siamese Region Proposal Networks (RPNs) within a Joint Detection-Association paradigm. Siamese trackers classify pixels as "target" or "background" before locating objects to track via CNN, reducing search space for high accuracy and real-time performance [61].

2.2 Perception Attacks

Perception Attacks against Object Detection. Figure 1 illustrates the attacks against OD and OT. There are three primary attacks against OD. Object deletion [60] (Figure 1-a) seeks to suppress one or more object detection results, causing the target to disappear. In this case, this could cause a vehicle collision with a pedestrian. Object misclassification (Figure 1-b) changes the class of one object to another object, such as a stop sign to a speed limit sign [10]. In this example, the autonomous vehicle may illegally speed up instead of coming to a stop, placing itself and others in danger. Lastly, object creation [37] (Figure 1-c) creates fake objects in the scene that OD sees as valid. This specific instance may cause an autonomous vehicle to halt unnecessarily.

Perception Attacks against Object Tracking. In addition to bringing greater accuracy, OT brings robustness against OD attacks. Thus, two major attacks have recently been put forward against OT itself. First, tracker hijacking attacks [24, 36] (Figure 1-d) move the bounding box of an object in a controlled direction. For example, against autonomous vehicles, an attacker can move the bounding box of an object in a victim vehicle's path to the side of the road. Thinking that the path is clear, the victim may continue its course and collide with the hijacked object. Second, cooling-shrinking [57] (Figure 1-e) targets OT to completely suppress all tracking results. For example, an attacker seeking unauthorized entry into a building with autonomous surveillance may leverage cooling-shrinking to remove their tracker and enter a restricted area undetected. Both attacks can be performed by using perturbations to shift the *attention* of OT, which causes it to search for objects in a different area. For tracker hijacking, the attention is moved to where the attacker desires the tracker to move [36], while for cooling-shrinking, the attention is set to *no* area, causing OT to abort searches for tracked objects [57].

2.3 Existing Countermeasures

2.3.1 OD Defenses

We describe three broad categories of OD defenses, focusing on why they fail for OT attacks.

Information Reinterpretation. These defenses alter the input data or model to provide additional context that can be leveraged to diminish the effect of adversarial perturbations. For example, defensive distillation [43] reconfigures the last layer of a deep neural network with a transformation computing the probability of a sample being a member of a given class, which provides additional robustness against misclassification attacks against OD. Meanwhile, feature squeezing [56] condenses samples with many different features into a single sample that is more robust to adversarial perturbations. However, both can be evaded by an adaptive attacker, e.g., by leveraging a CW attack [5].

Adversarial Input Detection. Another line of work detects and counters patch-based perturbations against classification-based convolutional neural networks (CNNs) [54]. These defenses also offer mathematical proofs that can guarantee a discrete level of performance against a given set of attacks. While effective for that domain, they are limited in scope to patch attacks and have only been demonstrated in the digital domain, whereas both OD and OT attacks have been affected by a wide variety of methods, digitally and in the real world.

Spatiotemporal and Contextual Consistency. Spatiotemporal and contextual consistency countermeasures are recent state-of-the-art OD defenses. Previous approaches leveraged clues about inter-object relationships [29], learned information about the environment [33], and/or extracted additional contextual information from the environment [16]. They ensure that the contextual information of the scene matches prior knowledge about object behavior (e.g., cars should not fly).

Current defenses, however, have only demonstrated efficacy against specific OD attack types: object misclassification [33] and object creation [37]. Because these works focus on attacks against OD, they do not consider attacks against OT that may not affect consistency. For example, a tracker hijacking attack can create an illusory lane-change movement on a vehicle driving forward. This motion is contextually consistent, but can cause a trailing autonomous vehicle to accelerate and hit the affected object [24, 36]. Additionally, these defenses suffer in situations where there is a lack of context to use for reasoning, e.g., in rural or low-light areas with very few objects. OT attacks such as cooling-shrinking [57] can take advantage of this by suppressing object tracking results and preventing them from being used in contextual reasoning.

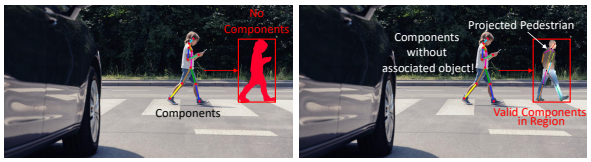
2.3.2 OT Defenses

Unlike OD attacks, OT attacks have different objectives than what is accounted for by current OD defenses. Current OT attacks alter the trajectory [24, 36] or suppress the tracking of objects [57] as opposed to changing an object's class or creating a new object. These attacks did not exist before the introduction of OT, and are understudied.

To our best knowledge, only one defense has been proposed against these attacks, utilizing noise cancellation to enhance tracking robustness [23]. However, while effective theoretically, this defense falls short in practice. Firstly, it relies on perfect projection of adversarial perturbations onto each frame, ignoring imperfections in the physical domain due to environmental conditions. Secondly, it assumes a constant attack presence, risking unintentional tracker hijacking when attacks are absent [36].

3 Threat Model

VOGUES is mainly designed to detect both tracker hijacking and cooling-shrinking attacks. Following the threat model



(a) (b)

Figure 2: (a) Component consistency under tracker hijacking. (b) A naive attempt to evade component consistency by projecting a fake pedestrian into the hijacked tracker's position.

common to these attacks, we assume *physical domain* attacks, where the attacker physically modifies the environment (via a projector [36]) or objects (via patches [33]) with adversarial noise. The attacker does not apply a norm-bound to the noise magnitude; instead, the strength of the perturbations is minimized to reduce the *attack cost*, which encapsulates three factors: stealthiness, required luminance to compete with other light sources, and the financial cost to the attacker (Section 7.4). Although we assume physical domain attacks, we also evaluate VOGUES against even more powerful attackers with *digital domain attacks* that remove real-world constraints (e.g., perturbations can be infinite). This puts a lower bound on the performance of VOGUES. However, we assume that the attacker is only able to manipulate input to the victim's sensors, and does not have access to the victim system's hardware.

We assess VOGUES against non-adaptive attackers not aware of VOGUES, and adaptive attackers aware of VOGUES and attempting to evade it while accomplishing attack goals. While VOGUES implicitly detects misclassification and deletion attacks (see Section 4), it cannot detect object creation attacks. This is because VOGUES is built to model human perception, and previous work has shown that creation attacks can look realistic to the human eye [37].

4 Motivation and Challenges

4.1 Rationale

VOGUES detects tracker attacks by enforcing the principle that every detected/tracked object must have corresponding components, obtained via the pose and orientation of the object, and vice versa. This follows from the principle that objects of interest generally consist of observable components that are arranged and move in expected ways. We add a further constraint that components must act consistently relative to each other and the object as a whole over *multiple frames.* This adds a layer of difficulty to adaptive attacks, as an attacker must simultaneously accomplish their original attack goal while altering the original object's components to maintain consistency.

Figure 2-a illustrates how VOGUES works against a tracker hijacking attack, which can lead to a failure to brake and a possible collision. Normally, an object's tracker and its components correlate with each other, being of similar size and in similar positions. However, the attack moves the tracker to a different location. To detect attacks, VOGUES compares extracted components with OT results to find *inconsistencies*. In this example, OT identifies the pedestrian in the attacker-specified location, but the pedestrian has no associated components, which is the first inconsistency. Meanwhile, VOGUES extracts components via pose estimation at the object's original location, but there is no tracker there, which is the second inconsistency. VOGUES can leverage the inconsistencies inherent in tracker hijacking, cooling-shrinking, and misclassification/deletion to offer a broad defense.

For this example, an attacker might take a naive adaptive approach and project a realistic-looking pedestrian into the hijacked tracker's position, as shown in Figure 2-b. In this case, the OT result has a component set, derived from the projected pedestrian, that it can be matched to. Yet, the hijacked pedestrian's original components still exist, and have no associated OT results. Thus, an inconsistency can still be found, and an alert can be raised. Adaptive attackers must suppress or change the original component results (e.g., via adversarial perturbations) to bypass VOGUES (See Section 7.4).

Inconsistency between objects and components manifests differently depending on the attack. For *tracker hijacking* attacks, the OT result and the original components remain unmatched, as they are forcefully separated from each other. For *cooling-shrinking* and *object deletion* attacks, stranded component sets without a parent object are the result, as the components remain when the objects are deleted. For *misclassification attacks*, though the component set and bounding box may occupy the same space, their *class* will be different, e.g., a car's components are seen for an object classified as a person.

4.2 Design Challenges

VOGUES models human perception by combining distinct neural networks based on human reasoning, which poses several challenges. These challenges stem from the interactions between the algorithms used, the difficulty in bridging their gaps with reasoning, and the ability of an adversary to adapt to detection methods they know are employed.

(C_1) Component Reconstruction Algorithm Selection. VOGUES is modeled after human perception, where an object's constituent components (e.g., the head, torso, and limbs of a pedestrian) should always be apparent. Additionally, components should behave consistently relative to each other and their parent object over a continuous timeframe However, the selection of the correct algorithms for this is nontrivial.

First, each component reconstruction algorithm should have orthogonal vulnerabilities compared to the OD/OT pipeline. Previous work demonstrated that models with different architectures, but similar vulnerabilities, can be exploited by transferring perturbations computed on one model to another [51]. Intuitively, using models based on

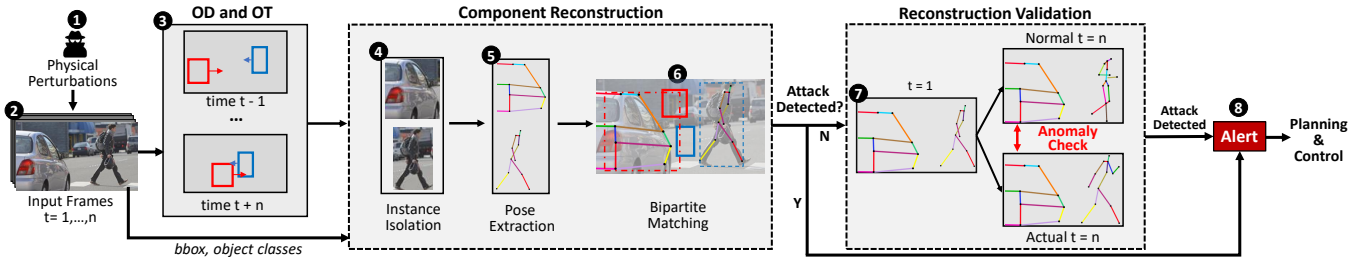


Figure 3: Illustration of VOGUES's stages.

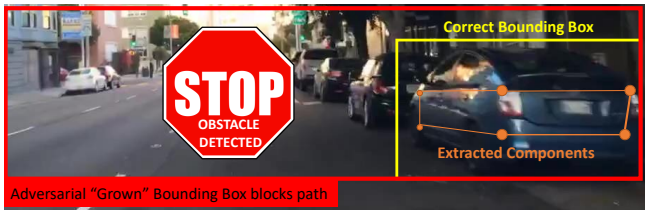


Figure 4: A tracker hijacking attack grows an object into a victim's path, evading an intersection-based consistency check.

different architectures and with different parameters makes it harder to create transferable/universal perturbations. We outline in Section 6.3 how using additional models that extract different features increases robustness, and we evaluate in Section 7.4.1 that forcing an attacker to exploit independent vulnerabilities greatly increases the amount of perturbations compared to dependent vulnerabilities.

Second, the algorithms must work in real-time. Domains such as AVs and video surveillance are time sensitive, with heavy consequences for delayed results, including collisions [36]. Therefore, while algorithms like Mask-RCNN [17] can classify objects at a pixel level in arbitrary detail, they are usually large networks that are difficult to scale. The typical running speed of < 10 fps achieved by most image segmentation algorithms [6, 17] will significantly delay results compared to the 10-30 fps accepted baseline [2].

(C_2) Consistency between Reconstruction and Perception. "Consistency" must be carefully defined to account for inherent error in ML predictions. For example, a naive approach demanding that the reconstructed details perfectly match the perception output 100% of the time would generate many false positives. Meanwhile, an overly flexible definition of consistency impairs the approach to detecting certain attacks.

As a metric, component consistency bridges the gap between ML algorithms in OD/OT and our component extraction algorithms. If consistency is defined in a way that does not reflect equivalent concepts in both perception and reconstruction, it can be evaded. For example, a naive reconstruction approach may use a simple intersection check to ensure the components of an object are within an object's bounding box. As illustrated in Figure 4, tracker hijacking could result in a bounding box on the side of the road to expand into a victim vehicle's path. As all the components of the vehicle on the side of the road are within the adversarial bounding box, a targeted vehicle may stop unnecessarily to avoid the incorrectly large adversarial object.

(C_3) Accounting for Adaptive Attacks. Given no restrictions on time, computational power, and perturbations, an adaptive attacker may succeed in achieving their goal. A trivial example is an attacker who causes an input video to darken in order to obscure all tracks. We note that with realistic constraints on the resources given above, such an attack is difficult.

Previous work has sought to address adaptive attackers by making it difficult to achieve the attack goal without a large number of perturbations [33]. This makes adaptive attacks highly visible and expensive to deploy using projector-based attacks (Section 7). However, recent advances in optimizing adversarial perturbation generation [5] have made effective countermeasures much more difficult to design.

5 VOGUES

In this section, we introduce how VOGUES defends against tracker hijacking, cooling-shrinking, object misclassification, and object deletion. Figure 3 presents the stages of VOGUES, which works as an addition to the perception pipeline of an AS. An attacker perturbs the environment, which is then picked up by the victim's video feed, to launch the attack against OT. The video feed is processed by the AS's perception pipeline, which outputs the bounding boxes and classes of every relevant object in the scene (❶-❸).

From here, VOGUES checks for *component consistency*, where the components, i.e., the key points forming an object, are checked for congruity with their parent object. The extraction of these components is split into three distinct steps to reduce mutual vulnerabilities with OD/OT attacks, addressing challenge C_1. In the first step, we perform *instance isolation*, performing a spatial transformation on the original input frames to enhance the accuracy of subsequent steps (❹). We next leverage the *pose estimation* algorithms to extract key points that form the "components" of each isolated object instance (❺). These components are then passed into *post-processing*, which uses non-max suppression

and pose-flow tracking to improve the component extraction results, before mapping them into *component sets*, each representing the components of one whole object, to be verified by *bipartite matching* against the supplied OD/OT results to ensure a 1-1 correspondence (❻).

We calculate the matching based on an Intersection over Union (IoU) metric, ensuring similar dimensions between components and their objects. More specifically, we define a bounding box for each component that covers their maximal bounds on the x-axis and y-axis, and compute their IoU with every other tracked object in the scene. Based on an application-specific IoU threshold, above which there is a significant similarity between an object and a component set, we first create a graph encoding potential associations between the object/set. We then conduct maximum bipartite matching on this graph to find if every object can be associated with exactly one component set and vice versa. This tunable spatial similarity metric for consistency addresses challenge $\mathtt{C}_2$.

If no bipartite matching solution is found, we alert the user and pass it to the AS's planning and control modules (❽). Yet, if a bipartite matching solution is found, and thus no attack is detected, we perform a final *reconstruction validation* check via a one-class LSTM anomaly detector trained to predict how the component sets should change over time (❼). The LSTM prediction, which takes the component sets of previous frames into account, is compared against the component data received via component reconstruction. If the reconstructed component data is substantially different from what the LSTM predicts, VOGUES raises an alert. Thus, an adaptive attacker must simultaneously complete their attack goal and hijack the components to realistically match that goal, addressing challenge $\mathtt{C}_3$.

5.1 Isolating Object Instances

To reduce interference from outside sources (e.g., other nearby objects of the same class), we first separate individual instances of the desired object class into tightly cropped samples. There exist a variety of computer vision techniques that can be used to accomplish this goal. For general object classes, we extend fully convolutional Region Proposal Networks (RPN) [28]. This is because RPNs learn transformations to apply to input video frames in order to isolate and crop specific object instances. The transformation-based process makes them suitable for applications where the target object may appear in a variety of configurations, positions, or rotations [25].

Given an input image, an RPN first uses feature mapping to generate a *heatmap* encoding of pixels likely to contain target object(s). We use this information for *region proposal*, which searches regions likely to contain targets in order to define bounding boxes per object. Lastly, both the heatmap and the region proposal results are merged via Region-of-Interest pooling by the RPN classifier, which returns the final instances of tightly-cropped isolated objects. This process is fully illustrated in Figure 12a in Appendix A.

On the other hand, for objects typically dispersed among crowds or at long distances, such as pedestrians, we leverage Spatial Transformer Networks (STNs), which perform better than RPNs under crowded/distant conditions [20]. A localization network first takes the input image and finds a set of affine transformation parameters (θ) to isolate the object instances. These parameters determine how the image should be altered in order to extract an individual object, e.g., how much scaling or rotation should be applied. Using θ, the original coordinate space G is converted into a sampling coordinate space T_θ, encoding the locations of the transformed image from which objects can be extracted via the 2D affine transformation:

$$T_\theta(G) = \begin{pmatrix} x_i^b \\ y_i^b \end{pmatrix} = [\theta_1\ \theta_2\ \theta_3] \begin{pmatrix} x_i^a \\ y_i^a \\ 1 \end{pmatrix}$$

where $\theta_1, \theta_2, \theta_3$ are vectors in $\mathbb{R}^2$, $\{x_i^b, y_i^b\}$ are the coordinates prior to the transformation, and $\{x_i^a, y_i^a\}$ are the coordinates after the transformation.

Lastly, to extract the cropped instances from the transformed coordinate space, a *sampler* converts grid segments within the image and transforms them into usable, tightly cropped instances. Figure 12b in Appendix A fully illustrates how we leverage the STNs for object isolation.

5.2 Pose Extraction

Once object instances are isolated, we extract the pose from the instances to use as components. The pose of objects is represented in the form of a graph, with nodes encoding the position and object class of user-defined key points within the object (e.g., joints for humans and bumpers for vehicles), and edges encoding the relationships between points (e.g., heads should be connected to necks).

To extract object poses, object instances are fed into a fully convolutional network (FCN) to yield a set of 2D *key points* describing visible details. For example, a vehicle's key points may consist of the left front wheel, the right side of the bumper, top, and bottom of the license plate, whereas a pedestrian's key points may consist of their arms, legs, spine, and head. Edges are then extrapolated based on class-based rules; for example, license plates should be connected to bumpers.

Given an image, the network minimizes the loss function:

$$L = \frac{1}{N}\sum_i (p_i - g_i)^2 + \log\left(\frac{e^{y_i}}{\sum_i e^{y_i}}\right)$$

where p_i is the predicted location of key point i, g_i is the keypoint's ground-truth location, and y_i is the ith position of the output layer. The pose extractor iteratively reduces the error in prediction p_i over a set number of attempts N.

At runtime, the pose extractor distills the image into feature sets, which are compared to the features of known, previously learned key points (e.g., the arm of a human or the license

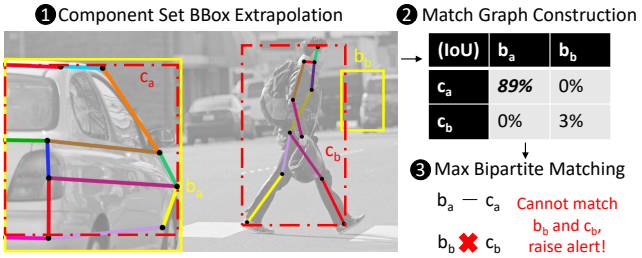


Figure 5: Our approach for component consistency checking.

plate of a car). The pose extractor produces a set of most likely predictions (each containing the proposed pose information of the object), along with a confidence score for each prediction, and returns the highest-scoring prediction as the final pose. This encodes all *visible* key points for an object, but not necessarily all the points: points that are occluded or otherwise obscured in a way that an object detector cannot cover (e.g., by deep shadow) are not returned.

5.3 Post-processing

Once poses for objects are extracted, we process the results to enhance the extraction accuracy and turn the pose information into usable component sets. For each object, we run non-maximum suppression [38] on all of the returned component sets. Specifically, we order component sets by their confidence scores, then eliminate any component sets that are within a class-specific *pose similarity distance* [11] to a higher-confidence set. This ensures that closely clustered, similar-looking component sets will be reduced to the single most confident set, which helps eliminate any duplicate component information (e.g., two sets returned for one pedestrian).

For objects extracted via STN, we map the pose information of each object back to the original coordinate space via a spatial de-transformer network (SDTN). The SDTN takes transformation vectors (θ_1, θ_2) and computes de-transformation vectors $(\gamma_1, \gamma_2, \gamma_3)$:

$$[\gamma_1\ \gamma_2] = [\theta_1\ \theta_2]^{-1}$$

$$\gamma_3 = -1 * [\gamma_1\ \gamma_2]\theta_3$$

These are used to invert the original transformation:

$$\begin{pmatrix} x_i^a \\ y_i^a \end{pmatrix} = [\gamma_1\ \gamma_2\ \gamma_3] \begin{pmatrix} x_i^b \\ y_i^b \\ 1 \end{pmatrix}$$

where (x_i^a, y_i^a) are the coordinates post-transformation and (x_i^b, y_i^b) are the coordinates pre-transformation.

To improve the accuracy of pose-extraction results, once the estimated poses are mapped back into the original coordinate space, we run pose-flow tracking [55]. Pose-flow tracking enforces spatio-temporal consistency in pose estimation between consecutive video frames, improving accuracy with minimal overhead to the component reconstruction process.

Once post-processing is complete, we aggregate the pose information of extracted classes and turn them into *component sets* to be used for consistency checking. Each component set represents a single object, and encodes the object's class, the image frame number, the absolute coordinates of each key point extracted by pose estimation, the class of each key point (e.g., hand, bumper), and their *relationships* to each other. The relationships between key points are created through class-based rules (e.g., hips should be connected to legs), and work for both complete key point sets and partially occluded ones. All final component sets found within a video frame are passed into the consistency-checking module to detect tracker hijacking and cooling-shrinking attacks.

5.4 Consistency Checking

We judge component consistency based on the similarity of dimensions between each component set and their corresponding tracking result. Figure 5 illustrates how our component consistency checking operates. For comparison against the original tracker bounding boxes (❶ in yellow, solid lines), we first extrapolate a bounding box from each component set by computing the maximum extent of the overall component set on the x- and y-axis (in red dashed lines).

We next create a *match graph* by examining the IoU between each tracker bounding box and each component set (❷). This forms the mathematical basis of our component consistency metric. The IoU computes the dimensional similarity between two bounding boxes, encoding the percentage that two bounding boxes overlap over their combined area. The less they differ in location, size, and shape, the higher the IoU. An IoU of 1 indicates that the two bounding boxes are identical. If the IoU is above a similarity threshold v, we construct an edge in the match graph (represented by bold text) between the relevant component set and the tracker bounding box, to represent potential 1-1 correspondence between them.

Lastly, we run maximum bipartite matching on the match graph to map every tracker bounding box to exactly one component set (❸). If a matching cannot be found, an inconsistency exists. In Figure 5, the bounding box b_b has been hijacked away from its component set c_b.

We formalize our consistency checking approach in Algorithm 1. We take, as inputs, the object bounding box sets b from the OD/OT pipeline, the component sets k from our reconstruction pipeline, and the acceptable similarity threshold $v \in [0, 1]$. The threshold v can be adjusted based on the inherent error in the constituent algorithms and depending on the domain (see Section 7.1). For example, if a security camera guards a large open area leading to a restricted space, a smaller, less strict v can be used, whereas autonomous vehicles can set v to ≥ 0.5 to accommodate the more time-sensitive nature of the application.

Algorithm 1 Component-Object Consistency Checker

Input: List of bounding boxes and corresponding classes $\texttt{b} = (\texttt{class} : \texttt{box})$, List of component sets and corresponding classes $\texttt{k} = (\texttt{class} : \texttt{components})$, acceptable similarity threshold $\texttt{v}$.
Output: Inconsistency indicator bit (`alert`).

```
1: function CHECK_CONSISTENCY(b, k, v)
2:     for class c ∈ b do
3:         if LENGTH(b[c]) ≠ LENGTH(k[c]) then
4:             return alert = 1                ▷ Unmatched components exist
5:         end if
6:         match_graph = {V = [b[c] ∪ k[c]], E = []}
7:         for box ∈ b[c] do
8:             for cmpnt ∈ k[c] do
9:                 cpt_box = MAX_EXTENTS(cmpnt)
10:                if box ∩ cpt_box/(box ∪ cpt_box) ≥ v then
11:                    match_graph[E].append(box ↔ cmpnt)
12:                end if
13:            end for
14:        end for
15:        bp_match_graph = ADD_SRC_SINK(match_graph)
16:        if FORD_FULKERSON(bp_match_graph) ≠ |k| then
17:            return alert = 1                ▷ Unmatched components exist
18:        end if
19:    end for
20: end function
```

Our consistency checking algorithm verifies each class separately in turn (Line 2), preventing objects and component sets from different classes from being matched together. Next, it ensures that the number of object bounding boxes and the number of component sets for a given class are equal (Lines 3-5). Here, inequality means that either one or more object bounding boxes have no components (indicative hijacking or misclassification), or one or more components have no parent object (indicative of cooling-shrinking or object deletion).

Assuming that there are an equal number of object bounding boxes and component sets, we initialize the match graph (Line 6) encoding which object may belong to which component set. We extrapolate each component set's 2-dimensional bounds (Line 9) to compare its location and size to OD/OT results. We draw an edge between an object and a component set if their IoU is greater than or equal to the acceptable similarity threshold v (Lines 10-12).

After the construction of the match graph, we run maximum bipartite matching between the components and the object bounding boxes to ensure that there is a one-to-one mapping. Specifically, we add a source node to all object bounding boxes in the match graph and a sink node to all component sets (Line 15). We then run the Ford-Fulkerson algorithm [12] to obtain the *maximum flow* of the match graph (Line 16). A maximum flow equal to the number of component sets/objects indicates that a one-to-one mapping exists. If such a mapping cannot be achieved, it means that at least one bounding box does not have a corresponding component, and vice-versa; therefore, we trigger an alert (Line 17).

6 Adaptive Attacks

Our component reconstruction pipeline is built to address previous OD and OT attacks. However, if an attacker is aware of VOGUES, they may attempt to adapt their attack to alter both the perception pipeline and VOGUES's consistency checking at the same time, disguising their attack. We detail our formulation on how such an adaptive attack might be launched, and introduce *reconstruction validation* to counter adaptive attacks.

6.1 Formulation

We create a new adaptive attack that aims to evade VOGUES while accomplishing the original attack goal. Although certain naive attacks can also be considered adaptive, e.g., wearing a camouflaged ghillie suit to hide a pedestrian, these methods are both conspicuous and capable of fooling human perception, which we consider outside of our threat model (Section 3). Thus, we extend the CW attack [5] to find the optimal minimized perturbations that avoid VOGUES's component reconstruction algorithms while still achieving the attack goal.

We minimize the perturbation amount with respect to the ℓ_2 *distance*, which takes the root-mean-square of the amount of perturbations applied to the image. We choose the ℓ_2 norm over ℓ_∞ because numerous small modifications to the image will generally lead to a smaller *maximum perturbation*, as opposed to a small number of large changes. A higher maximum perturbation increases the attack's visibility, required luminance, and cost to project (Section 7.4), thus previous work considers ℓ_2 the strongest metric to minimize adaptive attack perturbations [5,49]. Formally, we minimize the ℓ_2 norm $\|\delta\|_2$ for an image x to generate perturbation δ as follows:

$$\begin{aligned} &\text{minimize} \;\; \|\delta\|_2 + c \cdot f(x+\delta) \\ &\text{subject to} \;\; x+\delta \in [0,1]^n, \end{aligned}$$

where c is a tunable scaling constant that balances the importance between the two objectives [5], and $f(x+\delta)$ represents the attack success loss. Specifically,

$$f(x) = g(x) + \sum_i \left| p'_i(x) - p_i(x) \right|, \tag{1}$$

where $p(x)$ is the matrix of the predicted pose for x, and $p'(x)$ is a *target pose* for an object, which defines the targeted absolute coordinates that the targeted object's pose should be mapped. $p'(x)$ can be manually specified (e.g., an empty set meaning no pose for cooling-shrinking), or be derived from the original pose information (e.g., the same pose shifted 200 pixels left for tracker-hijacking), depending on the attack goal.

We use derived target poses for adaptive tracker hijacking, whereas we manually specify target poses based on the empty set for tracker deletion and cooling shrinking. We manually specify target poses for object misclassification attacks based on sample poses extracted from benign footage. Further, in

Figure 6: An adaptive attack hijacking the pedestrian's pose and the tracker. The component set dimensions (yellow, dotted line) are consistent with the hijacked tracker, but the pose is in an invalid shape and detected by reconstruction validation.

Equation 1, $g(x)$ represents the original objective function of the non-adaptive attack with constraints. For example, for tracker hijacking [36], $g(x)$ is given as:

$$\sum_{n=1}^{N} (L_c(I_n, p_c, \theta) - L_c(I_n, p_c^*, \theta)) + \lambda(L_r(I_n, p_r, \theta) - L_r(I_n, p_r^*, \theta))$$

where I_n is the proposal for input frame I at proposal n, λ is a fixed weight used to smooth ℓ_1 loss for regression, θ is the region proposal network's parameters, and N is the number of frames. Additionally, L_c and L_r are the OT classification and regression loss functions, and p_c/p_c^* and p_r/p_r^* encode the labels/pseudolabels for OT classification and regression.

We ensure with Equation 1 that both the original objective function $g(x)$ is achieved and that the achieved pose is as similar to the target pose as possible (and thus the difference between the two is 0). Using the CW objective function with the ℓ_2 optimization, the attacker finds the minimally visible amount of perturbations to accomplish the attack goal and evade VOGUES's detection. An adaptive attacker will continue to optimize their perturbations until a working solution is found. A solution is always possible given an infinite perturbation budget. However, finding the solution is not trivial, as the attack must simultaneously minimize Equation 1 and achieve the target pose. We quantify the difficulty of doing so theoretically in Section 6.3, and experimentally in Section 7.4.

6.2 Reconstruction Validation

Pose Hijacking. Previous work studied *pose hijacking*, which use adversarial perturbations to transform the correct pose into a target-specified pose anywhere in the image [21]. Our adaptive attack conducts pose hijacking to alter the victim's pose to match the attacked OD/OT data. Yet, it is difficult to hijack the entire pose of an object, as certain key points (e.g., the head and neck in humans) are more robust against attack than others [21]. Thus, hijacked poses tend to be unnatural and unrealistic, requiring extra perturbations to correct. We discuss the the behavior of poses under hijacking further in Appendix B.

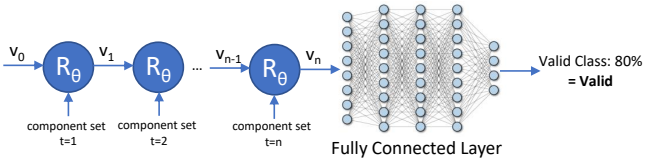


Figure 7: VOGUES's reconstruction validation remembers features (v) between frames, which are continually updated by the current component set via recurrent feedback loop (R_θ).

Validating Components. Leveraging the unusual appearance of hijacked poses, we develop a *reconstruction validation* pipeline against adaptive attacks. While an adaptive attacker may evade consistency checking, extracted components will likely have an unusual shape unless the attacker uses a much larger amount of noise (see Section 7.4.1), as illustrated in Figure 6. The hijacked pose (bounded by yellow, dotted lines) is consistent with the hijacked tracker bounding box. Thus, component reconstruction will not detect an attack. Yet, the pose indicates a flailing posture that should be impossible from a running start. We thus perform validation through anomaly detection against our component sets to detect adaptive attacks. With validation in place, an adaptive attacker's goal requires even less difference between the target and predicted poses; greater deviations can be picked up as invalid. We quantify how difficult this stricter goal is to achieve in the real world in Section 7.4.

We conduct reconstruction validation via a one-class LSTM network, similar to those used in anomalous network traffic detection [30]. The LSTM architecture is selected to learn order dependence in predicting the sequence of components [14]. For our system, we pass in a sequence of components across object classes, collected over the course of several video frames to allow for a complete analysis of the poses over time. Figure 7 illustrates the pipeline for reconstruction validation. Intuitively, the LSTM works by "remembering" the sequence of component sets of previous frames as it updates with the information from the current frame. Formally, it takes a sequence of features v for each frame over time, updating with current information via recurrent feedback loop:

$$v_i = R_\theta(f_i, v_{i-1}),$$

where v_i is the hidden feature vector calculated from previous feature vectors v_{i-1}, f_i is the raw feature vector at time i, and θ is a learned parameter identical for all recurrent operations.

Once all features are extracted at time n-1, the LSTM outputs the prediction of what the next component set should look like, taking into account the history of previous component sets. For example, if a pedestrian has been running beside an oncoming vehicle, the pedestrian will likely continue running in the next frame and not be suddenly at rest on the other side of the image.

This prediction is compared to the actual component information at time n, and passed into a fully connected

layer to return a validity score $\in [0,1]$, which encodes the likelihood that the component set at time n is compatible with the predictions of previous frames. This score is checked against a tuned hyperparameter threshold d, which is set based on our experiments (see Section 7.1). If a validity score of less than d is returned, the autonomous system is alerted to potential tampering with the component reconstruction pipeline. However, declaring components as valid does not preclude an alert from being thrown: the alert generated by consistency checking (Section 5.4) always takes precedence over the validation results.

6.3 Adaptive Attack Analysis

We provide a theoretical justification of the higher robustness of VOGUES against adaptive attacks due to the combination of multiple diverse models (ODT, pose estimation, and reconstruction validation). Previous work has shown that pose estimation models extract different, more detailed features from objects than ODT models [8]. With different extracted features, different errors occur when ODT models and pose estimation models are attacked (Section 7.3). The defensive value of utilizing additional models that have different errors has been examined by KEMLP [16], which utilizes additional task-specific models to integrate domain knowledge into OD to make it more robust. Although VOGUES works differently to KEMLP, providing independent results to be checked against ODT instead of feeding domain knowledge results directly into OD, our pose extraction system is a *permissive* type model [16] (i.e., if a pose is extracted for an object, a bounding box should be extracted as well), and our component consistency LSTM is a *preventative* type model [16] (i.e., given a set of poses relating to an object over time, these poses *must* move in a consistent manner). Thus, we can use KEMLP's Theorem 2 to describe the conditions where VOGUES will be more robust than ODT alone. We leverage the definition of a *truth rate* [16], which defines the rate at which a model gives results that are consistent with ground truth. Specifically, given an input distribution D containing both benign samples D_b and attacked samples D_a, the weighted accuracy of ODT can be described as a truth rate:

$$\alpha_* := \sum_{D \in \{D_b, D_a\}} \pi_D \alpha_{*,D}$$

where π_D is the probability distribution function for if a sample is in D_b or D_a, $\alpha_{m,I}$ is the truth rate at which a model m gives results that match ground truth for input samples I, and $*$ represents the ODT models.

Given this, the *combined* truth rate of the component reconstruction models $\mathcal{K}$ and the ODT models can be described. Let $\mathcal{K}, \mathcal{K}' \in \{I, J\}$, where I is the set of permissive models and J is the set of preventative models, and $\mathcal{K} \neq \mathcal{K}'$. Then:

$$\gamma_D := \frac{1}{n+1} \min_{\mathcal{K}} \left\{ \alpha_{*,D} - \frac{1}{2} + \sum_{k \in \mathcal{K}} \alpha_{k,D} - \sum_{k' \in \mathcal{K}'} \varepsilon_{k',D} \right\} \quad (2)$$

where n is the number of models represented by $\mathcal{K}$ and $\varepsilon_{m,D}$ is the *false rate* at which a model m gives results that do not match the ground truth. Intuitively, Equation 2 explains the accuracy of VOGUES, based on both ODT's accuracy and the accuracy/error rate of VOGUES.

According to Theorem 2 of KEMLP, by Chernoff bound on the true and false rates, as long as $\gamma_D > \sqrt{\frac{4}{n+1} \log \frac{1}{1-\alpha_*}} \quad \forall D \in \{D_b, D_a\}$, our defense can detect attacks against ODT. As a corollary, an adaptive attacker must increase their perturbations in order to lower γ_d to evade VOGUES and ODT. VOGUES achieves sufficient γ_D to detect attacks (Section 7.3), and we further support the theorem in Sections 7.4.1 to 7.4.3 with empirical data illustrating how real-world constraints significantly reduce an attacker's ability to find a solution that evades VOGUES.

7 Evaluation

Using both digital and real-world attacks, we evaluate VOGUES against non-adaptive attackers (Section 7.3) and adaptive attackers (Section 7.4) for autonomous driving and video surveillance applications. To evaluate successful detections by VOGUES, we consider only the scenarios where the given attack is already successful. Therefore, a "successful detection" is determined by whether VOGUES generates an alert for the successful attack. We also evaluate the false positive rate of VOGUES by measuring the number of alerts in benign scenarios, where no attacks are conducted (Section 7.5).

We evaluated our system on a laptop with a 9th Generation Intel i5-9300H processor, an NVIDIA GTX 1650 GPU, and 8 GB RAM running Python 3.8.10. With this setup, we achieved an average of 23 fps runtime, with 0.03s for instance isolation, 0.04s for component extraction, 0.0005s for matching, and 0.0007s for reconstruction validation.

VOGUES is able to detect 99.49% of digital attacks and 94.06% of physical attacks, with false positive rates of 1.78% in autonomous driving and 2.98% in video surveillance. Adaptive attacks require 49.9% stronger perturbations than non-adaptive attacks, significantly increasing the attack difficulty and making them impractical in the real world.

7.1 Implementation

We employ YOLOv3 [44] as our object detection platform and DaSiamRPN [61] as our object tracking platform, both of which have been used in recent previous work in object detection and tracking attacks [33, 36].

Component Reconstruction. We implemented all ML models in PyTorch 1.9.0cu102. For pedestrian component

reconstruction, we run a stacked hourglass network [11, 39] based on single-person pose estimation that extracts the pose information of each individual separately. For vehicle component reconstruction, we implement a real-time keypoint extraction model for vehicles, similar to approaches used for vehicle pose estimation and six-degrees-of-freedom 3D reconstruction of tracked objects [25, 26, 32, 46]. It is trained on the KITTI dataset [13] to extract up to 20 2D vehicle key points as pose information. We set our acceptable similarity threshold (v) to 0.5 for consistency checking.

Reconstruction Validation. To train the reconstruction validation LSTM model, we collected the output of our component reconstruction pipeline across 13000 videos in the UFC101 dataset [48] and 606 videos in the BDD100K dataset [59]. For each video, a sequence of components in each frame was encoded. We then trained the LSTM network to classify these sequences as valid component sequences. A randomly chosen 80% of the original data was used to train the LSTM, while the other 20% was used to validate the LSTM at each performance step. We set our hyperparameter validity score threshold (d) to 0.5, based on tuning via grid search in initial experiments.

Adaptive Attacks. For maximum compatibility with our existing model implementations, we adapted a PyTorch version of the original Carlini-Wagner attack [27] for adaptive experiments. Our modified version accepts target poses as inputs as opposed to classes and modified the constraints to account for both the original attack goal and evading VOGUES.

7.2 Evaluation Setup

7.2.1 Attacks

We evaluate VOGUES against previous attacks on the OD/OT pipeline. For digital attacks, we evaluate it against (a) tracker hijacking [36], (b) cooling-shrinking [57], (c) object misclassification [7], and (d) object deletion [19]. For physical attacks, we evaluate it against (a) tracker hijacking [36] and (b) object misclassification [7]. This is because, to our knowledge, they are the only two attacks against camera-based systems reliably demonstrated in the physical domain. We also modify these attacks with the formulation in Section 6.1 for adaptive attacks.

In these attacks, we focus on vehicle and pedestrian classes as they are the most safety-critical, causing physical injury and masked entry into restricted areas. Extending VOGUES for new object classes is straightforward, as discussed in Section 8.

7.2.2 Digital Datasets

We use two datasets to simulate physical attacks against our system in the digital domain: 197 videos from BDD100K [59] for autonomous vehicles and a curated subset of 19,562 images from MPII [1] for pedestrian video surveillance. We selected videos and images based on compatibility with the attacks evaluated, removing those with strong noise that prevents OD/OT from working. Together, these datasets cover a rich diversity in lighting conditions, driving environments, and human behaviors encountered in the real world. We conducted all four attacks, and their adaptive versions, against each video/image for a total of 19,759 samples per attack.

We implemented tracker hijacking attacks by moving trackers 200 pixels in the opposite direction of the victim object, e.g., moving the tracker of a left-of-center object to the right. For cooling-shrinking attacks, we removed all trackers from the image or video sample. For object misclassification attacks, we changed pedestrians to vehicles and vice versa. For object deletion attacks, we deleted the pedestrian or vehicle with the highest confidence from the OD output.

Table 1: VOGUES detection rate against non-adaptive attacks.

Attack Goal	Digital Attacks Pedestrian‡	Digital Attacks Vehicle‡	Real-world Attacks Pedestrian	Real-world Attacks Vehicle
Tracker Hijacking	99.49%	100%	100%	90.9%
Cooling-Shrinking	99.49%	100%	N/A†	N/A†
Misclassification	99.49%	100%	100%	91.67%
Object Deletion	99.49%	100%	N/A†	N/A†

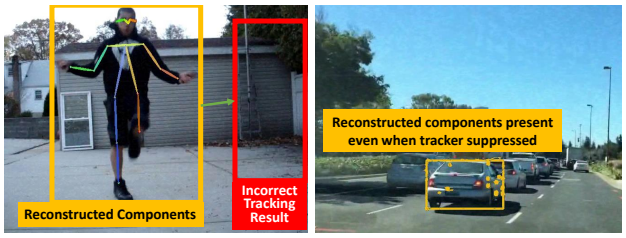
† Tracker-hijacking and misclassification attacks are conducted for real-world experiments as cooling-shrinking and object deletion have not been reliably demonstrated in the physical domain.
‡ Non-adaptive attacks do not affect component reconstruction, leading to identical detection performance regardless of digital attack method.

7.2.3 Real-world Physical Experiments

We conducted physical domain non-adaptive and adaptive attacks against moving vehicles and pedestrians using the same methods as previous work [18, 36]. For non-adaptive attacks, we collected 49 videos for successful tracker hijacking (33 for AVs and 16 for pedestrians) and 52 videos for successful misclassification attacks (36 for AVs and 16 for pedestrians). For adaptive attacks, we collected 43 videos each for tracker hijacking and object misclassification (33 for AVs and 10 for pedestrians). In these attacks, we chose videos in different locations and lighting conditions to represent as diverse an amount of variables and conditions as possible. We detail our attack setup and data in Appendix C.

In real-world tracker hijacking attacks, we aim to cause vehicle collisions or stoppages in AVs via tracker hijacking, or disguise entry into unauthorized areas in video surveillance. In misclassification attacks, we altered vehicles into pedestrians to cause incorrect AV planning & control decisions. To fool video surveillance, we misclassified pedestrians as other appropriate classes to ignore unauthorized entry.

We prioritized safety in real-world experiments by choosing low-traffic locations and times, engaging a third-party spotter, and obtaining permission from local police. We discuss our safety measures further in Appendix C.3.



(a) Tracker Hijacking (b) Cooling-shrinking

Figure 8: Detection against non-adaptive (a) tracker hijacking and (b) cooling-shrinking attacks in the digital domain.

7.3 Non-adaptive Attacks

Table 1 shows the detection success rate of VOGUES against non-adaptive attacks across digital and real-world domains. For digital attacks, we observed the same success rate regardless of attack type: 99.49% for pedestrians and 100% for vehicles. If an attack was detected in an image or video sample, VOGUES detected all attacks against that sample. Similarly, if an attack was not detected for a sample, VOGUES would always fail to detect the attacks against it. This indicates that the perturbations of non-adaptive attacks do not affect pose extraction performance.

In Figure 8, we illustrate two successful detections against OT attacks in the digital domain. In Figure 8-a, the pedestrian's components remain in their original place despite the tracker being hijacked to the right, raising an alert via consistency checking. In Figure 8-b, all tracking results are suppressed, but the components of a targeted vehicle are still present, again raising an alert via consistency checking.

In our real-world experiments, VOGUES detected 100% of tracker hijacking and misclassification attacks against pedestrians, and 90.9% / 91.67% of tracker hijacking / misclassification attacks against vehicles. Real-world attacks against pedestrians in restricted areas focus on attackers disguising their entry. Unlike digital domain attacks, these areas are typically well-lit and freer of crowds and occlusions, which enables VOGUES to extract components more precisely, improving the success rate.

However, overall we observe that VOGUES is slightly more successful in the digital domain than in the physical domain, with an average success rate of 99.49% vs. 94.06% in the real world. We attribute this to the larger sample size of digital attacks compared to physical attacks. In the real world, the detection failures were for very similar cases, such as misclassified vehicles that were distant and shrouded in darkness.

Figure 9 shows how VOGUES successfully detects physical-domain tracker hijacking attacks. On the left, the attacker hijacks the pedestrian's tracker to the left of the image to disguise their entry into an unauthorized area to the right. However, the pedestrian's components remain perceivable as they enter the restricted area. As they appear in distinct

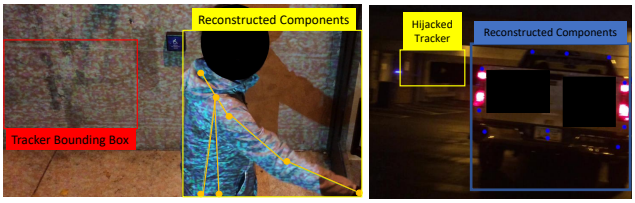
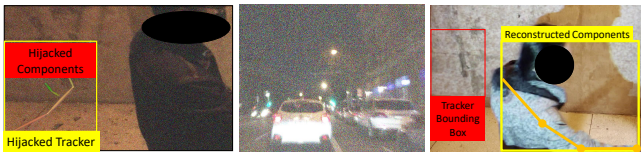


Figure 9: Two successful detections against non-adaptive tracker hijacking for physical domain video surveillance and autonomous driving.



(a) Without Validation (b) With Validation (c) Real World

Figure 10: Adaptive attacks (a) digitally applied against a real-world video, **without** reconstruction validation, causing an unnatural component reconstruction, (b) against a digital dataset **with** reconstruction validation, removing component reconstruction results entirely to conduct cooling-shrinking, and (c) in the real world, where adaptive attacks failed to fully suppress component reconstruction.

locations, this raises a component consistency alert. On the right, a tracker hijacking attack moves the truck's bounding box outside the path of the AV, attempting to induce a collision. However, the vehicle's reconstructed components remain in their original position, and an alert is raised.

7.4 Adaptive Attacks

We assess VOGUES against adaptive attackers (formulated in Section 6.1), which aims to simultaneously evade its component reconstruction and accomplish the attack goal.

7.4.1 Digital Attacks

In this set of experiments, we do not place any physical constraints on digital attacks, i.e., realistic perturbation regions. We compute the average per-pixel perturbation required to evade VOGUES successfully, quantifying the physical realizability of the attacks, rather than examining the attack success rate (which would be 100%). Below, we empirically demonstrate that increasing required perturbations also increases adaptive attack difficulty against VOGUES.

Without Reconstruction Validation. We measure the utility of reconstruction validation by comparing adaptive attack requirements with and without validation deployed. That is, without validation, the attacker simply specifies a *target zone* for the pose information (given as a bounding box) rather than a target pose. This method has looser success requirements,

as the attacker simply has to hijack the pose into the correct region without worrying about a valid shape over time. In this case, any choice of target pose successfully evades VOGUES.

We find that, across all attacks, the average change in perturbation amount between non-adaptive and adaptive attacks *without* reconstruction validation is $21.91\% \pm 4.92\%$. Figure 10-a illustrates the visual impact of such an attack. Both the tracker and the pedestrian's components have been hijacked to the left of the image, away from the restricted area. Although the pose is consistent, it is in an unnatural shape, with several joints missing and the remaining limbs unnaturally stretched. The looser success requirements lead to fewer required perturbations, which are certainly more visible than the almost imperceptible noise in Figures 8 and 9. However, employing validation makes the required perturbations even stronger.

With Reconstruction Validation. With reconstruction validation, the attacker must now achieve a target pose that evades our one-class LSTM network. Although one or more solutions always exist, and our adaptive attack will choose the target pose that has the least required perturbations to achieve, reconstruction validation ensures that a large perturbation amount is required to evade detection, which is conspicuous and difficult to achieve in real-world settings (Section 7.4.3).

Specifically, the average change in perturbation between non-adaptive and adaptive attacks *with validation* is $49.9\% \pm 3.3\%$ across all attacks, approximately doubling the without-validation amount. This comes with only a 0.05 fps increase in running time, making it feasible to employ. The average per-pixel difference ($|I - I_p|$) between the perturbed image and the original image amount is 127.27 ± 8.43, with a max difference of 255 (black areas of the image become white). Overall, the required perturbation for each sample depends on the brightness of the environment, with the least bright sample (averaged over all pixels) having the lowest change in perturbation of 33.14% between non-adaptive and adaptive attacks, and successively brighter samples requiring further perturbations to prevent being washed out.

Figure 10-b shows the visual impact of adaptive attacks under reconstruction validation, where an adaptive cooling-shrinking attack removes all trackers and component sets. Although both object detection and pose extraction have had their results completely suppressed, the required perturbations are visually striking.

Previous user studies [21] for visibility of similar pose-hijacking adversarial perturbations showed that, above a maximum $||I - I_p||_\infty$ value of 32, over 97.84% of participants are able to clearly see the perturbations. This increases the chance of an AV operator taking manual control over their vehicle, or a pedestrian notifying authorities and interrupting the attack.

7.4.2 Constrained Digital Attacks

To quantify the effect of the surrounding layout on the success of adaptive attacks, we constrain the perturbations through LiDAR-based attack zone generation [36]. Applying this approach to DriveTruth [35], an autonomous driving data generator built on the CARLA simulator, we create a dataset with identified physically perturbable regions. This allows us to generate attacks that perturb invalid areas, e.g., the sky and bright lights in an environment. Our final dataset contains 40 videos with randomized environmental factors, including road traffic, victim speed, and traffic light timing.

Under constraints enforced on the perturbable regions, the adaptive attack success rate drops by 15%. 10% of attacks are detected and 5% of attacks fail to achieve their goals. The failure cases are due to the limited perturbable regions in a sample, which are smaller than the successful cases.

These results show that the layout of the environment influences whether or not an adaptive attack can work in the physical world, even when we assume the projector's strength is infinite and unaffected by any light source.

7.4.3 Real-world Attacks

In this set of experiments, we constrain noise generation to physically perturbable regions for physical domain adaptive attacks; however, unlike digital attacks, the projector no longer has infinite perturbation strength. This makes the projected perturbations more susceptible to environmental lighting conditions, overpowering and washing out the adversarial noise [35]. As a result, VOGUES detects 100% of the successful physically launched adaptive attacks.

Figure 10-c shows an example of a physically launched tracker-hijacking attack. The adaptive noise suppresses the component reconstruction elements, such as the neck and much of the torso. However, the effect is incomplete, and the remaining unaffected reconstructed components are enough for VOGUES to detect the inconsistency and generate an alert.

Projectors with higher power can generate the required adaptive perturbations in the physical domain, but they are also more expensive. To estimate the projector perturbation cost [33], we calculate the luminance required for projected noise with ℓ_∞, ambient luminance, and distance square, i.e., $\text{Lumen} \propto \ell_\infty(\Delta) \cdot \text{Illum} \cdot d^2$.

We observed an ambient luminance of 301x, and our $\sim$200 projector was able to effect perturbations at $\ell_\infty = 0.1$ at a distance of two meters. Scaling to the required ℓ_∞ of 1 for adaptive attacks, an attacker would need a 9K-lumen projector, typically priced at \$13K [9], and in daytime lighting conditions, where ambient illuminance is 40 klx, a 75K-lumen projector would be required, which is priced around \$375K [3].

Table 2: Comparison of VOGUES's performance with other OD defenses at detecting **object misclassification**, changing stop signs into pedestrians in the BDD100K dataset. We note that none of the other systems are able to detect tracker hijacking and cooling-shrinking attacks on OT.

Method	Detection Rate	FPR
VOGUES	98.48%	1.78%
PercepGuard [33]	99%	5%
SCEME [29]	81.67%	8.33%
KEMLP [16]	93.75%	0%

7.5 False Positive Rate Analysis

We evaluated VOGUES on 1,600 benign samples from the BDD100K dataset and 1,442 samples from the MPII dataset, for a total of 3,042 samples. Because no attacks were conducted for this set of experiments, we counted a false positive for every sample that VOGUES generated an alert.

Across all samples, VOGUES has a total false positive rate of 3.29%, 2.98% for video surveillance, and 1.78% for autonomous driving. 8% of FPs were caused by errors in the victim's object detection and tracking pipeline rather than errors in VOGUES. For example, in cases where an object is obscured by motion blur, the object detection and tracking pipeline fails to lock onto the object while the reconstruction pipeline perceives it, creating an inconsistency between components.

All other FPs were due to component reconstruction errors, such as extracting two or more component sets for one object. This can occur in uneven lighting conditions or when the object is obscured by the environment. As computer vision advances and pose estimation models improve, component reconstruction will be able to better handle these edge cases.

8 Discussion and Limitations

Comparison of VOGUES with Existing Countermeasures. To our knowledge, there are currently no practical defense or detection approaches against OT attacks. However, several state-of-the-art defenses for OD exist, which aim to mitigate misclassification attacks in the AD domain. These defenses do not work against tracker hijacking and cooling-shrinking attacks. Specifically, using the BDD100K dataset [59], we evaluated PercepGuard [33], SCEME [29], and KEMLP [16] against both attacks, and all three defenses had a 0% detection rate. Our tracker hijacking attacks, moving a vehicle 200 pixels to the left or right, emulated real-world lane changes, while our cooling-shrinking attacks, removing suppressing all OD/OT results from the start of the attack, emulated empty environments without objects. Both scenarios are contextually consistent under previous OD defenses.

However, previous defenses were built for the *OD* domain, whose attacks have different characteristics. In Table 2, we compare these defenses with VOGUES against *misclassification* attacks on OD, changing the stop sign class into the pedestrian class. We note that VOGUES is not mutually exclusive with any of these defenses, and we encourage using OD-specific defenses alongside VOGUES for the best coverage against different attacks.

We found that KEMLP [16] yields a 0% FPR, with its first-order logic approach less susceptible to FPs compared to approaches relying on neural networks, including VOGUES. However, KEMLP's detection rate is 4.73% lower than VOGUES's, requiring more precise application-specific knowledge rules to detect attacks. In contrast, VOGUES does not need application-specific knowledge since component consistency principles apply to all objects with extractable components. PercepGuard [33], similarly, yields a 0.52% higher detection rate than VOGUES, but its FPR is 3.22% higher because it can detect certain benign movement behaviors (e.g., aborting a lane change by swerving back into the original lane) as attacks. Unlike PercepGuard, VOGUES relies less on temporal features, such as movement, to detect attacks and more on the appearance of objects. Lastly, SCEME yields a 16.81% lower detection rate and 6.55% higher FPR than VOGUES. Unlike VOGUES, SCEME is heavily affected by the richness of object context and performs poorly when there are fewer objects in a scene (e.g., in foggy or dark driving conditions).

Extending Object Classes. We demonstrated the effectiveness of VOGUES on the pedestrian and vehicle classes as they are the most critical object classes in safety-critical autonomous driving and video surveillance. Here, we study how VOGUES can extend to other classes and domains.

To make VOGUES work for arbitrary classes, users first need to create appropriate component definitions for each new object class. Once components are defined, the other algorithms of VOGUES work without any additional implementation. For example, Figure 11 shows how a DriveTruth [35] dataset is used to define components for a traffic sign or traffic light. We use the semantic LiDAR data to define a traffic sign's components as a hexagon encompassing the sign's corners, or a traffic light's components as its four corners and active light. Once the class's general shape is defined, it can be automatically fitted to semantically segmented data to train a neural network model for extracting components, as we described in Section 5.2.

However, we note that it is more challenging to define components for certain classes of objects with irregular, unique patterns, such as trees. Evaluations on a generalized VOGUES without class-specific post-processing shows that it has 0.99% higher attack detection rates in the real world, albeit with a 0.81% higher FPR (Appendix D). Future work can extend experiments to assess VOGUES in rare scenarios, such as extreme weather conditions.

VOGUES as a Full Defense. VOGUES alerts the system when an attack is detected, but does not take any automated recovery actions, such as issuing commands to an AV to mitigate or prevent the attack. It is challenging to anticipate all the

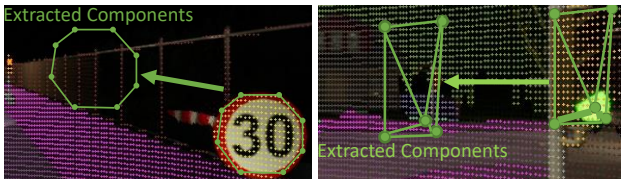


Figure 11: Example of how VOGUES generalizes to other object classes: Components extracted from a traffic sign (left) and a traffic light (right).

consequences of attacks in different application domains. By outputting alerts, VOGUES gives the user or system the flexibility to decide how to respond optimally to an attack.

We plan to extend VOGUES into a complete defense by implementing a continuous scoring system, e.g., using the IoU between objects and their components, to allow ASs to make robust decisions. To prevent false positives from impeding the operation of an AS, alerts from VOGUES can be used as an indicator or confidence value to be used by a higher-level, decision-making security system. For example, in the AV domain, the victim system can adopt more cautious driving behavior in response to a single isolated alert. For consecutive alerts, the victim vehicle might yield control to a human driver. Similarly, in video surveillance, a single alert might notify a human security guard, while consecutive alerts should directly sound the alarm for unauthorized entry. As the accuracy of the perception and component reconstruction algorithms improve, VOGUES can be updated to have a lower FPR.

9 Conclusions

We introduce VOGUES, the first practical countermeasure against tracker hijacking and cooling-shrinking attacks. VOGUES models the human principle that any visible object must also have tangible constituent components. VOGUES is also effective against OD attacks, including misclassification and object deletion. VOGUES leverages pose estimation to reconstruct the key components of objects in a video frame. To enhance robustness against adaptive adversaries, we propose an LSTM validator to detect adaptive attacks against the reconstruction framework. We evaluated VOGUES against adaptive and non-adaptive attackers, using digital datasets alongside data collected in the real world. VOGUES successfully detects 99.49% of attacks in the digital domain and an average of 93.81% in the physical domain.

Acknowledgments

We thank our shepherd and the anonymous reviewers for their valuable suggestions. This work has been partially supported by the National Science Foundation (NSF) under grant CNS-2144645, and the Army Research Office (ARO) under grant W911NF2110320. The views expressed are those of the authors only. We would like to thank Chenyi Wang for his insights on the writing and formal notation.

References

[1] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *Computer Vision and Pattern Recognition*, 2014.

[2] Gaurav R Bagwe. Video frame reduction in autonomous vehicles. In *Michigan Technological University (Thesis)*, 2018.

[3] Barco. Xdl-4k75. `https://www.10kused.com/product/barco-xdl-4k75-lhjz-52035/`. [Online; Accessed 27-August-2023].

[4] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *IEEE International Conference on Image processing (ICIP)*, 2016.

[5] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, 2017.

[6] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *European Conference on Computer Vision (ECCV)*, 2018.

[7] Ka-Ho Chow, Ling Liu, Margaret Loper, Juhyun Bae, Mehmet Emre Gursoy, Stacey Truex, Wenqi Wei, and Yanzhao Wu. Adversarial objectness gradient attacks in real-time object detection systems. In *IEEE International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications*, 2020.

[8] Xiao Chu, Wei Yang, Wanli Ouyang, Cheng Ma, Alan L Yuille, and Xiaogang Wang. Multi-context attention for human pose estimation. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[9] Epson. Pro l1490u. `https://tinyurl.com/444ynkej`. [Online; Accessed 15-October-2023].

[10] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physical-world attacks on machine learning models. *arXiv preprint arXiv:1707.08945*, 2017.

[11] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. RMPE: Regional multi-person pose estimation. In *International Conference on Computer Vision (ICCV)*, 2017.

[12] Lester Randolph Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 1956.

[13] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.

[14] Alex Graves. Supervised sequence labelling. In *Supervised sequence labeling with recurrent neural networks*. Springer, 2012.

[15] Lie Guo, Linhui Li, Yibing Zhao, and Zongyan Zhao. Pedestrian tracking based on camshift with kalman prediction for autonomous vehicles. *International Journal of Advanced Robotic Systems*, 2016.

[16] Nezihe Merve Gürel, Xiangyu Qi, Luka Rimanic, Ce Zhang, and Bo Li. Knowledge enhanced machine learning pipeline against diverse adversarial attacks. In *International Conf. on Machine Learning*, 2021.

[17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *International Conference on Computer Vision*, 2017.

[18] Shahar Hoory, Tzvika Shapira, Asaf Shabtai, and Yuval Elovici. Dynamic adversarial patch for evading object detection models. *arXiv:2010.13070*, 2020.

[19] Hao Huang, Yongtao Wang, Zhaoyu Chen, Zhi Tang, Wenqiang Zhang, and Kai-Kuang Ma. Rpattack: Refined patch attack on general object detectors. In *International Conf. on Multimedia and Expo*, 2021.

[20] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 2015.

[21] Naman Jain, Sahil Shah, Abhishek Kumar, and Arjun Jain. On the robustness of human pose estimation. In *Computer Vision and Pattern Recognition Workshops*, 2019.

[22] X. Ji, Y. Cheng, Y. Zhang, K. Wang, C. Yan, W. Xu, and K. Fu. Poltergeist: Acoustic adversarial machine learning against cameras and computer vision. In *IEEE Symp. on Security and Privacy*, 2021.

[23] Shuai Jia, Chao Ma, Yibing Song, and Xiaokang Yang. Robust tracking against adversarial attacks. In *Euro. Conf. on Computer Vision*, 2020.

[24] Yunhan Jia Jia, Yantao Lu, Junjie Shen, Qi Alfred Chen, Hao Chen, Zhenyu Zhong, and Tao Wei Wei. Fooling detection alone is not enough: Adversarial attack against multiple object tracking. In *International Conference on Learning Representations (ICLR)*, 2020.

[25] Lei Ke, Shichao Li, Yanan Sun, Yu-Wing Tai, and Chi-Keung Tang. Gsnet: Joint vehicle pose and shape reconstruction with geometrical and scene-aware supervision. In *Euro. Conf. on Computer Vision*, 2020.

[26] Hanme Kim, Stefan Leutenegger, and Andrew J Davison. Real-time 3d reconstruction and 6-dof tracking with an event camera. In *European Conference on Computer Vision*, 2016.

[27] kkew3. Rich-documented pytorch implementation of carlini-wanger's l2 attack, 2018. [Online; Accessed 15-October-2022].

[28] Daniel Konig, Michael Adam, Christian Jarvers, Georg Layher, Heiko Neumann, and Michael Teutsch. Fully convolutional region proposal networks for multispectral person detection. In *Computer Vision and Pattern Recognition Workshops*, 2017.

[29] Shasha Li, Shitong Zhu, Sudipta Paul, Amit Roy-Chowdhury, Chengyu Song, Srikanth Krishnamurthy, Ananthram Swami, and Kevin S Chan. Connecting the dots: Detecting adversarial perturbations using context inconsistency. In *European Conference on Computer Vision*, 2020.

[30] Yanmiao Li, Yingying Xu, Yankun Cao, Jiangang Hou, Chun Wang, Wei Guo, Xin Li, Yang Xin, Zhi Liu, and Lizhen Cui. One-class lstm network for anomalous network traffic detection. *Applied Sciences*, 2022.

[31] Shaoshan Liu, Jie Tang, Zhe Zhang, and Jean-Luc Gaudiot. Computer architectures for autonomous driving. *IEEE Computer Magazine*, 2017.

[32] Javier García López, Antonio Agudo, and Francesc Moreno-Noguer. Vehicle pose estimation via regression of semantic points of interest. In *Intl. Symp. on Image and Signal Processing and Analysis*, 2019.

[33] Yanmao Man, Raymond Muller, Ming Li, Z. Berkay Celik, and Ryan Gerdes. That Person Moves Like A Car: Misclassification Attack Detection for Autonomous Systems using Spatiotemporal Consistency. In *USENIX Security Symposium*, 2023.

[34] Garima Mathur, Devendra Somwanshi, and Mahesh M Bundele. Intelligent video surveillance based on object tracking. In *International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE)*, 2018.

[35] Raymond Muller, Yanmao Man, Z. Berkay Celik, Ming Li, and Ryan Gerdes. DriveTruth: Automated autonomous driving dataset generation for security applications. In *International Workshop on Automotive and Autonomous Vehicle Security (AutoSec), collocated with NDSS*, 2022.

[36] Raymond Muller, Yanmao Man, Z. Berkay Celik, Ming Li, and Ryan Gerdes. Physical Hijacking Attacks against Object Trackers. In *ACM Conference on Computer and Communications Security (CCS)*, 2022.

[37] Ben Nassi, Yisroel Mirsky, Dudi Nassi, Raz Ben-Netanel, Oleg Drokin, and Yuval Elovici. Phantom of the adas: Securing advanced driver-assistance systems from split-second phantom attacks. In *ACM SIGSAC conference on computer and communications security*, 2020.

[38] Alexander Neubeck and Luc Van Gool. Efficient non-maximum suppression. In *International Conference on Pattern Recognition (ICPR)*, volume 3, 2006.

[39] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, 2016.

[40] Andreas Nussberger, Helmut Grabner, and Luc Van Gool. Aerial object tracking from an airborne platform. In *international conference on unmanned aircraft systems (ICUAS)*, 2014.

[41] Milan Ondrašovič and Peter Tarábek. Siamese visual object tracking: A survey. *IEEE Access*, 9, 2021.

[42] Optoma. Optoma lv130 ultra-portable projector. `https://www.optoma.com/vn/product/lv130/`, 2021. [Online; Accessed 11-October-2022].

[43] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy (S&P)*, 2016.

[44] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[45] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[46] Héctor Corrales Sánchez, Antonio Hernández Martínez, Rubén Izquierdo Gonzalo, Noelia Hernández Parra, Ignacio Parra Alonso, and David Fernandez-Llorca. Simple baseline for vehicle pose estimation: Experimental validation. *IEEE Access*, 8, 2020.

[47] Md Kamruzzaman Sarker, Lu Zhou, Aaron Eberhart, and Pascal Hitzler. Neuro-symbolic artificial intelligence: Current trends. *AI Communications*, 2021.

[48] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. A dataset of 101 human action classes from videos in the wild. *Center for Research in Computer Vision*, 2012.

[49] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *International Conference on Learning Representations (ICLR)*, 2013.

[50] Waymo Team. Utilizing key point and pose estimation for the task of autonomous driving, 2022.

[51] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *Intl. Conf. on Learning Representations*, 2017.

[52] Christian Wittpahl, Hatem Ben Zakour, Matthias Lehmann, and Alexander Braun. Realistic image degradation with measured PSF. *Autonomous Vehicles and Machines*, 2018.

[53] Chong Xiang, Arjun Nitin Bhagoji, Vikash Sehwag, and Prateek Mittal. Patchguard: A provably robust defense against adversarial patches via small receptive fields and masking. In *USENIX Security Symp.*, 2021.

[54] Chong Xiang, Saeed Mahloujifar, and Prateek Mittal. {PatchCleanser}: Certifiably robust defense against adversarial patches for any image classifier. In *USENIX Security Symposium*, 2022.

[55] Yuliang Xiu, Jiefeng Li, Haoyu Wang, Yinghong Fang, and Cewu Lu. Pose Flow: Efficient online pose tracking. In *The British Machine Vision Conference (BMVC)*, 2018.

[56] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *Network and Distributed System Security (NDSS)*, 2017.

[57] Bin Yan, Dong Wang, Huchuan Lu, and Xiaoyun Yang. Cooling-shrinking attack: Blinding the tracker with imperceptible noises. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2020.

[58] Zhuolin Yang, Zhikuan Zhao, Boxin Wang, Jiawei Zhang, Linyi Li, Hengzhi Pei, Bojan Karlaš, Ji Liu, Heng Guo, Ce Zhang, et al. Improving certified robustness via statistical learning with logical reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[59] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving video database with scalable annotation tooling. *Computer Vision and Pattern Recognition Conference (CVPR)*, 2020.

[60] Yue Zhao, Hong Zhu, Ruigang Liang, Qintao Shen, Shengzhi Zhang, and Kai Chen. Seeing isn't believing: Towards more robust adversarial attack against real world object detectors. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2019.

[61] Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *European COnference on Computer Vision (ECCV)*, 2018.

Appendix

A RPN and STN Isolation

VOGUES uses two methods to isolate object instances, which are selected to account for different conditions the objects may appear in. RPNs (Figure 12a) learn how to generate feature maps based on an input image. The maps can be used to regress the tightly cropped object instance. This is ideal for targets that can appear from many different angles and positions, such as vehicles. Meanwhile, Spatial Transformer Networks (Figure 12b) regress scaling and plane rotation parameters (θ) to better isolate objects that are among crowds or at long distances, such as pedestrians.

B Pose Hijacking

Our adaptive attack seeks to "hijack" the pose estimation results of component reconstruction to match VOGUES's results with the adversarial attack goal. However, attacks against pose estimation often result in unnatural poses, with limbs or the spine bent at odd angles. Figure 13 demonstrates pose hijacking attempts against two common HPE models, attention [8] (Figure 13-a) and 8-stacked [39] (Figure 13-b).

In these attacks, the attacker aims to hijack the original pose (in orange) into the pose of a man sitting (in red, on top of the original pose). Different model architectures result in slightly different behavior: for example, attention HPEs may experience less displacement in the overall pose, but they may fail to detect certain limbs. Meanwhile, stacked HPEs may have a greater pose displacement, creating a warped shape as individual key points within the pose are shifted independently. No matter the architecture, the difference in resistance to perturbation between key points means that exponentially stronger perturbations are required for a full pose hijacking. According to user studies, the required perturbations can be detected 97.84% of users [21].

C Real-World Attack Details

C.1 Attack Setup

For tracker hijacking attacks, we used an Optoma LV130 mini projector [42] to project adversarial perturbations onto the environment. Against AVs, we place the projector on the dashboard of the victim vehicle, while against surveillance, we place the projector directly behind our security camera to effect perturbations within its field of view.

For misclassification attacks against AVs, we mounted a 40-inch LCD monitor to the back of a vehicle via a tailgate hitch mount and used the monitor as a dynamic adversarial patch. The vehicle was driven normally, but extra caution was taken when making sharp turns to prevent the mount from swinging.

In our real-world experiments, we used an HD camera to record footage in 1920x1080 resolution. For vehicle-based experiments, the camera was mounted directly above the dashboard to simulate the camera of an autonomous vehicle. For surveillance experiments, the camera was placed high on a wall looking downward at a building entrance to simulate a security camera watching a restricted area.

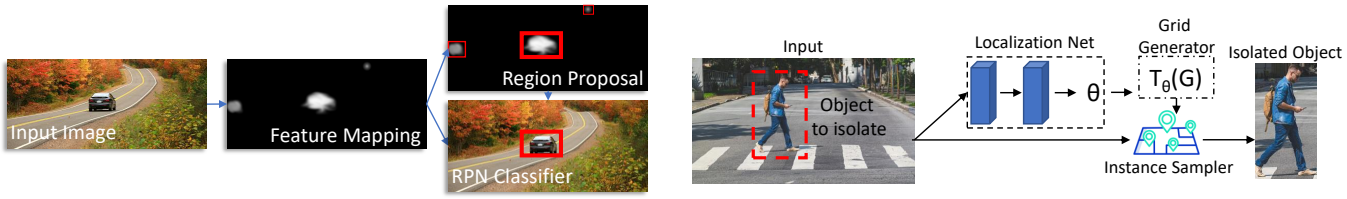
C.2 Video Selection

In our evaluation of real-world non-adaptive attacks against AVs, we have three extra videos for misclassification attacks compared to tracker hijacking attacks. This is due to three more classification attacks against AVs succeeding than tracker hijacking. For both adaptive and non-adaptive attacks, we removed videos that were too similar to other samples. This was done to reduce repetitive videos that may artificially boost detection rates by having the exact same structure. No removal was necessary for the AV domain, as variations in traffic, pedestrians crossing roads, and other environmental factors created a large variety in collected videos. Yet, in the surveillance domain, with a static camera observing the same position, more similarity between videos was encountered, resulting in fewer pedestrian examples than AV examples.

C.3 Safety in Real-World Experiments

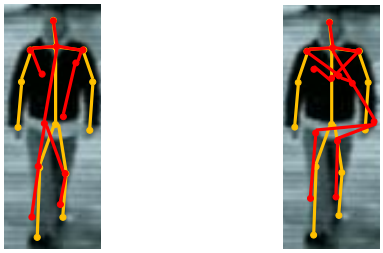
We made safety our highest priority during real-world experiments. We obtained permission to conduct our experiments from the Special Services Division of the local police department, and patrol officers were notified before the experiments. We carefully selected empty locations with sufficiently little traffic, as determined by a third-party spotter who continually supervised conditions to ensure safety. For monitor experiments, we secured the LCD monitor via bungee cord to absorb shock from sharp movements and prevent it from swinging around the environment.

During all vehicle experiments, we drove on roads where the speed limit was 25 MPH, and we did not exceed 20



(a) (b)

Figure 12: VOGUES's process of isolating objects with (a) Region Proposal Networks and (b) Spatial Transform Networks.



(a) Attention HPE (b) 8-stacked HPE

Figure 13: Results of adversarial attacks on hourglass pose estimation (HPE) models (with original pose beneath in orange) in video surveillance domains.

Table 3: VOGUES's attack detection performance with/without class-specific post-processing.

Mode	Digital Attacks	Real-World Attacks	FPR
Class-specific	99.49%	94.06%	3.29%
Generalized	99.49%	95.05%	4.1%

MPH when driving. We avoided aggressive acceleration and braking and obeyed traffic rules. Experiments were paused every 5 minutes in order to check the equipment and ensure it had not come loose or unstable.

All experiments were conducted with three experienced personnel at a time: an attack equipment operator, a victim equipment operator, and a third-party spotter to supervise the experimental conditions. All 3 maintained constant communication via phone during the experiments. The experiments were supervised and approved by professors experienced in autonomous vehicle research and experiments. In line with standards set by previous works [33, 36, 37], we conducted experiments without direct exposure to sunlight to reduce the effects of ambient lighting conditions on the effectiveness of the tested attacks.

D Extension to Other Classes and Domains

D.1 Ablation Study on General Pipeline

To comprehensively evaluate the effectiveness of VOGUES on general classes, we evaluate VOGUES with both class-specific post-processing, such as pose-flow tracking for pedestrians, and with a general pipeline where all classes conform to the minimum specifications outlined in Sections 5.1 to 5.3. The results are described in Table 3. Overall, we observe identical performance in digital attacks and a 0.99% greater attack detection rate for the generalized system than the class-specific system. However, this comes at a 0.8% higher false positive rate for the general system. The general system is more aggressive in outputting detection alerts because the pose results are no longer corrected through tracking. The lack of correction generates incorrect component reconstructions more often, whether or not an actual attack is performed. These results suggest that although the general specifications for all classes can achieve promising performance, victims may choose to enhance the pipeline with class-specific post-processing for more accurate results, depending on their requirements.

Figure 14: A common false positive case in sports tracking, where a crowd of people not in camera focus will be extracted as one bounding box by OD but have multiple components extracted. These cases are uncommon in the safety-critical domains of autonomous driving and surveillance due to the use of focus-free cameras.

D.2 Non-Safety Critical Domains

We evaluated VOGUES's performance in the sports tracking domain, where a camera automatically tracks the movements of athletes to keep them in focus. Sports tracking is less safety-critical compared to autonomous driving and surveillance but contains very different behaviors. Across 782 sports tracking examples from the MPII dataset, VOGUES obtains a 100% attack detection rate, but also a 20.2% false positive rate.

Figure 14 illustrates such an example of a common sports tracking-related false positive. An out-of-focus crowd is extracted as a single object by object detection, but multiple objects are seen by component extraction. Autonomous vehicle and surveillance applications typically use focus-free cameras [52], and do not suffer from focus-related blurring issues. Sports tracking can also contain more contorted poses that are more difficult to extract than poses from other applications. As pose estimation models are improved, component reconstruction may better handle more difficult domains.