Robotics: Science and Systems 2024
Delft, Netherlands, July 15-July 19, 2024

Dynamic Adversarial Attacks on Autonomous Driving Systems

Amirhosein Chahe Chenan Wang Abhishek Jeyapratap Kaidi Xu Lifeng Zhou
Drexel University
{ac4462, cw3344, aj928, kx46, lz457}@drexel.edu

***Abstract*—This paper introduces an attacking mechanism to challenge the resilience of autonomous driving systems. Specifically, we manipulate the decision-making processes of an autonomous vehicle by dynamically displaying adversarial patches on a screen mounted on another moving vehicle. These patches are optimized to deceive the object detection models into misclassifying targeted objects, e.g., traffic signs. Such manipulation has significant implications for critical multi-vehicle interactions such as intersection crossing, which are vital for safe and efficient autonomous driving systems. Particularly, we make four major contributions. First, we introduce a novel adversarial attack approach where the patch is not co-located with its target, enabling more versatile and stealthy attacks. Moreover, our method utilizes dynamic patches displayed on a screen, allowing for adaptive changes and movements, enhancing the flexibility and performance of the attack. To do so, we design a Screen Image Transformation Network (SIT-Net), which simulates environmental effects on the displayed images, narrowing the gap between simulated and real-world scenarios. Further, we integrate a positional loss term into the adversarial training process to increase the success rate of the dynamic attack. Finally, we shift the focus from merely attacking perceptual systems to influencing the decision-making algorithms of self-driving systems. Our experiments demonstrate the first successful implementation of such dynamic adversarial attacks in real-world autonomous driving scenarios, paving the way for advancements in the field of robust and secure autonomous driving.**

I. Introduction

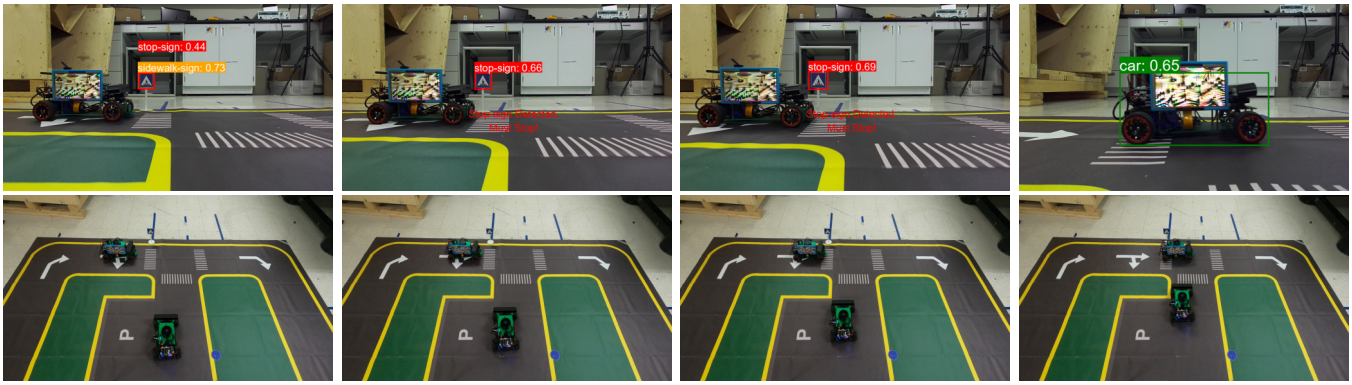
As autonomous vehicles and driver assistance systems become more prevalent, it is imperative to ensure that their decision-making systems are robust. The ability to detect and recognize traffic signs and cars accurately is crucial for autonomous vehicles to make safe and efficient decisions [35, 9, 37]. Deep neural networks (DNNs) [32] provide an efficient framework for real-time object recognition widely used by autonomous driving systems. However, it is not immune to adversarial attacks, which can potentially have serious consequences in real-world scenarios [33, 24, 39, 42]. In the past several years, scientists have investigated the reasons behind neural networks' vulnerability to adversarial examples and devised techniques for generating digitally perturbed images [8, 19, 27, 22, 5]. In a range of real-world scenarios, several studies indicated that adversarial perturbations on actual objects can deceive DNN-based object detectors in real-world scenarios [36, 13, 7, 3, 2, 40, 28, 4, 17]. For instance, To avoid detection, an individual can either carry a cardboard plate with a printed adversarial patch as described in [34] or wear an adversarial T-shirt as proposed by [40], both oriented towards the surveillance camera. Furthermore, [6] and [30] have demonstrated the feasibility of real-world adversarial attacks on Yolo [25] and Faster R-CNN [26]. They focus on manipulating stop signs by hiding the stop sign using an adversarial stop sign poster or by affixing adversarial stickers onto the sign.

However, in [20], it showed that the printed patches cannot fool object detectors across a wide range of distances and angles. To come up with a variety of distances and angles, we aim to deceive and mislead the DNN detectors by generating dynamic adversarial patches which are adapted based on the camera and target positions. We place the patches in a location distinct from the location of the target object. Specifically, we display the patch using a screen mounted on a robot car rather than on the target traffic sign itself. Our methodology employs an optimization process that operates at the pixel level, aiming to display a patch that significantly enhances the probability of misclassifying the object within a substantial dataset. Notably, our attack is carried out by a dynamic patch displayed on the screen, making its formulation and simulation more challenging due to several environmental variables.

The goal of this work is to affect the decision-making of an autonomous vehicle by affixing a custom-designed dynamic patch to another moving vehicle, causing the object detector model to misclassify an object, e.g., a non-restrictive traffic sign as a restrictive one as shown in Figure 1. This adversarial attack could potentially manipulate the behavior of the attacked vehicle in various vehicle interaction scenarios including navigating intersections, changing lanes, and merging into a lane, underpinning the importance of robust object detection in safety-critical autonomous driving systems.

Contributions. In this paper, we make four main contributions as follows.

- We formulate the problem of attacking the object detection of traffic signs by displaying a dynamic patch on a screen mounted on an autonomous robot car which enables the patch not only to be dynamically changing and moving but also short-lived and harder to detect.
- We reformulate the loss function of state-of-the-art adversarial neural network models by integrating a positional loss term that aligns the target object bounding box to increase the likelihood of a successful attack in the presence of corroborating sensory data.
- We propose SIT-Net, which simulates the color and contrast transformation of the displayed image on the



(a) Adversarial patch attacks on the Pedestrian sign.

(b) Benign trial.

Fig. 1: Sequential frames of a multi-robot interaction in an intersection: The top row of (a) demonstrates a dynamic adversarial patch attack, where the patch is displayed on the screen of the `patch car`. The patch is designed to mislead the `camera car`'s perception system, i.e., the camera car thinks the Pedestrian sign is a Stop sign. The top row of (b) illustrates the system's correct behavior during a benign trial without adversarial interference. The bottom rows of (a) and (b) are the top views of the settings.

screen as captured by the camera, to mitigate the gaps between the simulation and the physical world.

- We conduct extensive experiments to show successful attacks across various multi-vehicle interaction scenarios passing the intersection and attacking various traffic signs including Go-straight, Turn, and Pedestrian signs. This allows the attacker to adversarially manipulate other vehicles' decisions to take the lead and move with lower risk.

Furthermore, to the best of our knowledge, *we are the first to design and demonstrate dynamic adversarial attacks on real-world autonomous driving systems*.

II. Related Work

In the field of autonomous vehicles, the authors in [43] proposed systematic solutions for Hiding Attack (HA) and Appearing Attack (AA), enhancing the robustness of adversarial examples (AEs) against varying distances and angles. In [14], a method for generating physical AEs that fool traffic sign recognition of autonomous vehicles was presented which employs single and multiple bounding box filters, as well as considering different attack vectors, such as hiding, appearance, non-target, and target attacks. In [18], the authors proposed Short-Lived Adversarial Perturbations (SLAP) that allow adversaries to create physically robust real-world AEs using a projector that provides better control over the attack compared to traditional adversarial patches, as projections can be turned on and off as needed and leave no obvious trace of an attack. Furthermore, in [12], a dynamic adversarial patch technique was introduced to circumvent the object detection model by employing several dynamic patches applied to a target object that adapts to the location of the camera. The authors showed by switching between optimized patches based on the camera's position, the attack can adapt to different situations and achieve optimal results.

In all the previous physical targeted attacks, the patch is deployed at the location of the target object, which makes the patch implementation hard and or impossible in most practical cases. In this paper, we focus on the AEs in which a patch is in a different place from the target object and is dynamically displayed through the screen mounted on a either moving or stationary vehicle. Moreover, the final goal is to mislead the autonomous driving system's decision-making instead of purely deceiving its perceptions.

III. Problem Formulation

We investigate physical dynamic adversarial attacks on a DNN object detector for various autonomous driving scenarios where two robot cars, namely `patch car` and the `camera car`, are interacting with each other. The `camera car` employs a DNN object detector to detect other cars and traffic signs, which informs its decision-making process. Additionally, it utilizes a range of sensors, such as Lidar, to acquire the positions of these objects. The `patch car` is equipped with a screen that displays a deceptive patch. We aim to trick the `camera car`'s decision making such as coming to a halt by attacking its object detector, thereby allowing the `patch car` to take over and navigate through with earlier passing time and lower risk. Importantly, we also consider the camera car's ability to cross-reference detected object positions with its proximity data. For example, the agent matches detected objects' locations with the Lidar point clouds to verify them for the decision-making process [38, 1]. This adds an extra layer of complexity to the task of successfully deceiving it. In the following, we introduce our attack model in two dimensions—the attacker and the objective.

A. Attacker

Attacker, i.e., the `patch car`, is a specialized agent that can display adversarial patches dynamically on a display screen attached to it. Instead of altering objects physically, this agent aims to manipulate object recognition processes through visual perturbations shown on its screen. The attacker employs a white-box attack strategy, granting it full access to the object detector model. It can also obtain the proximity of the objects including the position of the `camera car` using a Lidar.

To formally define the dynamic patch, let $\mathbb{S}$ be the state space representing all possible positions of the `camera car`, s_c and the `patch car`, s_p, and $\mathbb{P}'$ be the space of all possible adversarial patches. The function T maps the patch δ given a tuple of states (s_c, s_p) to a specific transformed adversarial patch δ' in $\mathbb{P}'$. Moreover, to model environmental parameters affecting the patch visibility, we integrate the variable $\phi \in \Phi$ into the map function. This can be denoted as:

$$T(\delta, s_c, s_p, \phi) : \mathbb{P} \times \mathbb{S} \times \mathbb{S} \times \Phi \rightarrow \mathbb{P}' \tag{1}$$

The variable ϕ represents a specific color transformation of the displayed patch in the camera's image over all possible ones. We discuss it in the Section IV-B.

B. Objectives

We aim to devise an attack that misguides the object detector model, causing it to incorrectly classify a traffic sign to a target sign when viewed from the perspective of the `camera car`. We formally define the problem given that:

- $\mathbb{X}$: the space of all possible input images.
- $\mathbb{Y}$: the space of potential output predictions.
- $\mathbb{S}$: the space of possible positions of the cars.
- $f : \mathbb{X} \times \Theta \rightarrow \mathbb{Y}$: the detector model, parameterized by weights $\theta \in \Theta$.

The objective is to find the patch that maximizes the probability of the target class, e.g., stop-sign.

$$\begin{aligned} \underset{\delta}{\text{maximize}} \quad & P[f(\pi(x, \delta, s_c, s_p), \theta) = (\boldsymbol{p}_{\text{target}}, \boldsymbol{b}_{\text{target}})], \\ \text{subject to} \quad & \pi(x, \delta, s_c, s_p) = x + T(\delta, s_c, s_p, \phi) \\ & \forall x \in \mathbb{X},\ \forall s_c, s_p \in \mathbb{S},\ \forall \phi \in \Phi. \end{aligned} \tag{2}$$

$\pi(x, \delta, s_c, s_p)$ denotes the input image after applying the transformed patch. The patch must be applied within the coordination of the display screen which depends on the positions of the cars. $T(\delta, s_c, s_p, \phi)$ denotes the adversarial patch displayed on the screen transformation function provided earlier in Equation 1. $\boldsymbol{p}_{\text{target}}$ denotes the output prediction corresponding to the confidence of the target. $\boldsymbol{b}_{\text{target}}$ denotes the output prediction corresponding to the bounding box of the target.

We utilize gradient descent optimization to identify the patch that maximizes the misclassification rate of the detector model. Gradient descent is widely used for training adversarial attacks on DNNs [16]. In our case, we aim to optimize the patch that minimizes the loss function, as described in the section III-C.

According to the proposed function 1 for the patch, it depends on the position of the cars (s_p, s_c) and the environmental parameters. In Section IV-A, we introduce how to capture the changes in the positions of the cars and integrate these, resulting in a dynamic patch. To integrate the environmental parameters Φ, we propose a solution in Section IV-B.

C. Objective Function

To effectively deceive the `camera car`, our approach must address both the positional accuracy and the classification accuracy of the adversarial patch. The objective function $\mathcal{O}(\delta)$ is defined as follows:

$$\begin{aligned} \mathcal{O}(\delta) &= \mathbb{E}\left[\boldsymbol{p}_{\text{target}}(i) \cdot \boldsymbol{p}_{\text{obj}}(i) \cdot \text{IoU}(\boldsymbol{b}_{\text{target}}(i), \boldsymbol{b}_{\text{orig}})\right], i \in \mathbb{M} \\ \delta^* &= \min_{\delta} -\mathcal{O}(\delta). \end{aligned} \tag{3}$$

$\boldsymbol{p}_{\text{target}}(i)$ is the probability of the i-th bounding box being the target class (e.g., stop sign). $\boldsymbol{p}_{\text{obj}}(i)$ is the objectiveness probability of the i-th bounding box which is a likelihood that measures the confidence that the bounding box contains an object. The $\text{IoU}(\boldsymbol{b}_{\text{target}}(i), \boldsymbol{b}_{\text{orig}})$ is the Intersection over Union (IoU) between the i-th bounding box of the target class (e.g., stop sign). The $\boldsymbol{b}_{\text{target}}(i)$ is predicted by model and $\boldsymbol{b}_{\text{orig}}$ is the original object bounding box. $\mathbb{M}$ is the set of all the bounding boxes overlapping the original object. δ^* is the optimized patch. The objective function, $\mathcal{O}$, is designed to optimize two key factors:

Confidence score. The function aims to increase the probability that the object detector misclassifies the objects to the target class (e.g., a stop sign). This involves manipulating the patch in such a way that the object detector is highly confident in its misclassification.

Bounding box overlap. Alongside the confidence score, the function also aims to maximize the overlap between the

bounding box predicted by the object detector caused by the adversarial patch and the bounding box of the actual object. The combination of these two factors ensures that the adversarial patch not only deceives the detector in terms of classification but also aligns accurately with the expected position of the target object, as determined by the `camera car`'s other sensory inputs, e.g., Lidar.

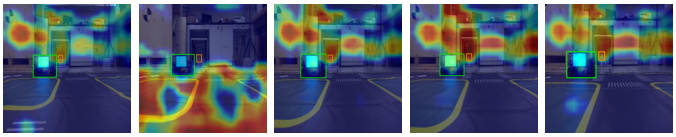


Fig. 2: The position of objects in an image influences the performance of adversarial patches. The influence, as visualized in heatmaps, shifts according to changes in objects' positions. The color of a heatmap reflects the influence level (red to purple denoting most to least influence).

IV. Approach

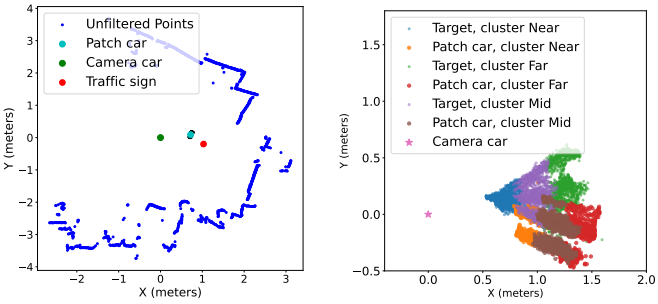
This section elaborates on our proposed approach for generating dynamic adversarial patches. Section IV-A explains the method for collecting patch training data that captures the dynamics of the environment. In Section IV-B, we demonstrate the image transformation of the patch that is displayed on the screen. Finally, we describe the patch optimization in Section IV-C.

A. Dynamic Environment Settings

The output of the object detection models (e.g., the regular yolov5s [15]) such as cars or traffic signs can be greatly influenced by the objects' position in the image. This positioning directly impacts the model's focus or "attention", altering the detection outcomes based on how far or near these objects are positioned relative to the camera's perspective. To visualize the principal components of the learned features and representations, we apply Eigen-CAM [23, 11] which reflects the influence level of areas of an image on the model through a heatmap (red to purple denoting most to least influence). As depicted in Figure 2, the model's attention shifts according to changes in object positions. This in turn impacts the effectiveness of adversarial patches that are not overlaid on the target object. To compensate, a dynamic patch that adapts to the target's position could be a more effective approach. Without direct access to the `camera car`'s captured images during an attack, the `patch car` can estimate the target's position and adjust its patch accordingly to account for positional changes.

1) Data collection: We employ Lidar to obtain the positions of the cars and the target object. Multiple autonomous driving scenarios involving the `patch car` and the target object are collected synchronously using Lidar and camera data.

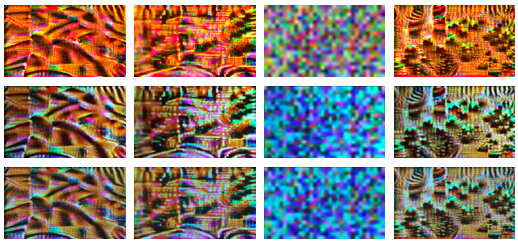
Image and Lidar recording. Both the `patch car` and the target object are recorded using a camera, and their positions are recorded via Lidar. This synchronized dual recording



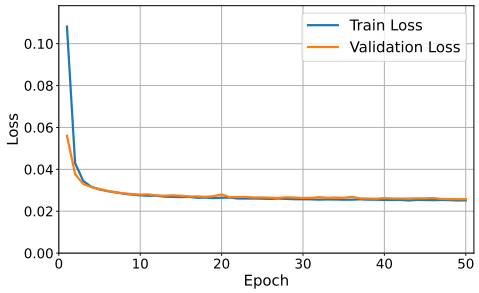
(a) The coordinates of the cars and the target in one data frame where the `camera car`'s position is the origin.

(b) Clustered 2D position points of the `patch car` and the target where the `camera car`'s position is the origin.

Fig. 3: Analysis of data clustering based on positions: (a) Lidar map showing the coordinates of the cars and the target in a single frame; (b) Cluster plot of the `patch car` and the target for all frames in a dataset.



(a) SIT-Net predictions. The first row shows input images (X), the second row shows camera-captured images (Y), and the third row shows the model's predictions.



(b) Loss over training epochs for SIT-Net.

Fig. 4: SIT-Net predictions and the loss during its training.

setup is designed to capture a variety of scenarios, including those where either one or both subjects are in motion. We enhance our position tracking accuracy using Lidar-based SLAM techniques, specifically AMCL [10], which affords robust pose estimation even in dynamic and unpredictable environments. Figure 3a demonstrates the `camera car`'s map generated with lidar points.

Screen contour identification. A full blue image is displayed on the screen during the recording sessions. This practice assists in the accurate identification of screen contours, which is crucial for the perspective transformation of the patches during the training phase.

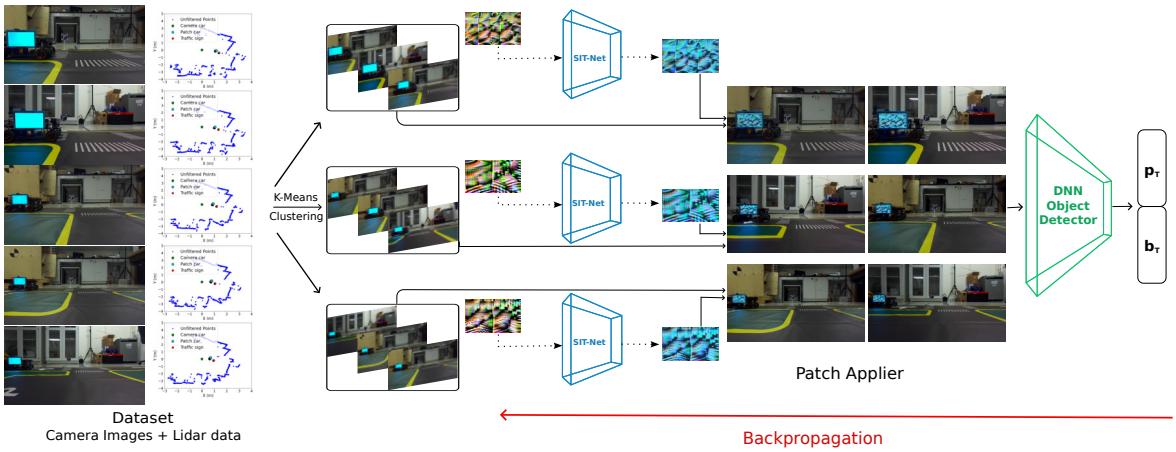


Fig. 5: The patch optimization pipeline.

2) Data clustering: The recorded data is then clustered by the K-means clustering algorithm based on the distance of the `patch car` and target object from the `camera car`. This way, the positions of the `patch car` and the target in the images in the same cluster are within the same threshold. Each cluster represents a specific range of distances, and a dedicated patch is trained for each cluster. Figure 3b shows the clustered points of the `patch car` and the target. This enhances the performance of the patches that are optimized for different clusters. Appendix A provides a more detailed overview of our dataset.

Dynamic patch display. The trained patches are then employed in a dynamic fashion. The system utilizes the `camera car`'s position to determine the appropriate patch to display, ensuring a more effective and contextually relevant visual transformation.

This structured and strategic approach to data collection and processing ensures that our system is capable of handling a variety of real-world scenarios, making the patches dynamic and enhancing the overall effectiveness of the visual transformations.

B. Screen Image Transformation Network (SITNet)

To model the visual changes of the patch in the images captured by the camera, we use a convolutional neural network (CNN) model. The SITNet is a CNN designed to transform the colors and contrast of images to mitigate the difference between simulation and the real world. It specifically focuses on adapting the appearance of images displayed on a screen, as captured by the camera, ensuring a more accurate representation and smoother transition between the simulated and physical environments. SITNet comprises two convolutional layers. In the following, we describe data processing and loss functions in the SIT-Net training steps.

1) Screen Data Collection and Prepossessing: We manually collected the dataset used to train SITNet to address the challenges of discrepancies between simulated and real-world visual data. The collection procedure is as follows:

Input image generation. A diverse set of patch-like images is created, ensuring a variety of color and contrast scenarios. These patches are of the same size as the targeted screen area to maintain consistency.

Screen display. The generated images are displayed on a screen, capturing a wide range of total variations to adequately represent different potential real-world conditions.

Image recording. A camera is utilized to record the displayed images. Precise coordination of the four corner points of the screen area is maintained to aid in the subsequent image-processing steps.

Image preprocessing. The captured images are subjected to cropping and perspective transformation based on the known screen coordinates. This step ensures that the images are accurately aligned and standardized, mirroring the conditions of the screen display.

This comprehensive data collection and preprocessing procedure ensures that the network is exposed to a wide spectrum of visual variations, enhancing its ability to generalize and accurately transform images in practical scenarios.

2) Loss Function and Optimization: The mean squared error is used in the loss function to reduce the difference between predicted and real-world images. However, these pixel-level differences might not correlate well with how images are perceived or their similarity. Perceptual loss addresses this limitation by incorporating features extracted from pre-trained neural networks that are designed to mimic human visual perception. Moreover, a total variation term [21] is added to the loss function to smooth the predicted images.

Mean squared error (MSE) loss. We quantify the pixel-wise difference between the predicted and real-world images by MSE loss.

$$\mathcal{L}_{\text{MSE}}(\hat{Y}, Y) = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2 \tag{4}$$

where N is the total number of pixels in the image.

Perceptual loss. The perceptual loss measures the difference in style between images at the feature level rather than the pixel level. The first nine layers of the pre-trained VGG19 network [29] are used to extract feature maps from the input and target images. Then the MSE between these feature maps is computed. This way, the model is forced to generate images that are perceptually similar to the target images. The perceptual loss is defined below.

$$\mathcal{L}_P(\hat{Y}, Y) = \frac{1}{M}\sum_{j=1}^{M}(V(\hat{Y})_j - V(Y)_j)^2, \tag{5}$$

where M is the total number of elements in the VGG feature maps.

Total variation (TV) loss [21]: The TV loss below encourages spatial smoothness in the predicted image.

$$\mathcal{L}_{\text{TV}}(\hat{Y}, Y) = \sum_{i,j}\left|\hat{Y}_{i,j+1} - \hat{Y}_{i,j}\right| + \left|\hat{Y}_{i+1,j} - \hat{Y}_{i,j}\right|. \tag{6}$$

The overall loss is a weighted sum of MSE loss, Perceptual loss, and TV loss.

$$\mathcal{L}(\hat{Y}, Y) = \mathcal{L}_{\text{MSE}}(\hat{Y}, Y) + \alpha\mathcal{L}_P(\hat{Y}, Y) + \beta\Delta\mathcal{L}_{\text{TV}}(\hat{Y}, Y). \tag{7}$$

The network is trained over $n_{\text{epochs}} = 50$ epochs, with a learning rate of 0.001. The weights for the perceptual and total variation losses are set as $\alpha = 0.02$ and $\beta = 0.01$, respectively. Figure 4 depicts the training curves and the SIT-Net predictions for an environment instance.

Algorithm 1 Adversarial Patch Optimization with SITNet

Require: Set of clustered images $\mathbb{X}_{\text{clustered}}$, original traffic sign bounding box $\boldsymbol{b}_{\text{orig}}$, patch $z \in \mathbb{P}'$, learning rate η, number of iterations N, top-k value k, SITNet model M_{SITNet}
Ensure: Set of optimized patch $\mathbb{P}^*$

1: Initialize z
2: **for** x in $\mathbb{X}_{\text{clustered}}$ **do**
3: **for** $i = 1$ to N **do**
4: $z' \leftarrow M_{\text{SITNet}}(z)$
5: $\pi(x, s_c, s_p) = x + \delta(s_c, s_p, \phi)$
6: $Y = f(\pi(x, s_c, s_p), \theta)$
7: $\mathbb{M} \leftarrow \text{Filter}(Y, \boldsymbol{b}_{\text{orig}})$
8: $\mathbb{M}_k \leftarrow \text{Top-k}(\mathbb{M})$
9: $\text{IoU}_k \leftarrow \text{IoU}(\mathbb{M}_k, \boldsymbol{b}_{\text{orig}})$
10: $\mathcal{O} \leftarrow \frac{1}{k}\sum_{j=1}^{k}\left(\text{Conf}_{\mathbb{M}_k[j]} \times \text{IoU}_{k[j]}\right)$
11: $z \leftarrow z + \eta \cdot \nabla_z \mathcal{O}$
12: **end for**
13: $\mathbb{P}^* \leftarrow z$
14: **end for**
15: **return** $\mathbb{P}^*$

C. Patch Optimization

The optimization procedure for the patch is described in Algorithm 1. We optimize a patch for each cluster in the dataset. First, the patch is transformed by the SIT-Net. Then we apply the transformed patch, z', to the images and pass it to the object detector. The line 7 includes filtering of the object detector's prediction and extraction of all bounding boxes in the area of the original object. The objectiveness probabilities of these bounding boxes are integrated into the class probabilities. Then we extract the top-k target class confidence values. Subsequently, in line 9, the IoU of the top-k target class bounding boxes with the original object bounding box is computed. In the next step, the IoU values are multiplied by the top-k confidence, and its mean is calculated. This value represents the alignment and confidence of the selected bounding boxes. Finally, this value is utilized in the optimization process of the patch. Figure 5 shows a pipeline of the patch optimization process. In summary, this process begins with the clustering of the images through the K-Means method using LiDAR data, then the patches that are transformed by the SIT-Net are applied to images of each cluster. The patches are optimized by backpropagating.

V. Evaluation

We test our proposed dynamic attack in different scenarios where two robotic agents interact with each other and must follow traffic signs.

A. Experimental Setup

Agents. We use two Rosmaster R2 robot cars with Ackermann structure [41] equipped with Lidar and IMU sensors for localization and mapping. The `camera car` uses a Zed2 camera [31] for image recording. The `patch car` is equipped with a seven-inch LCD screen attached to its side to display the patches. The car's decision-making planner employs object detection results to plan for its motions. Object detection and patch training are carried out by a server using a NVIDIA RTX4090 GPU.

Object detector model. For each robot car, we fine-tuned the model from yolov5 [15] to detect traffic signs and the other robot car. The model trained over 1.8 K images with the weights from a pre-trained model yolov5s and the input image size of 640×640. Our customized dataset and code are available through GitHub.[1]

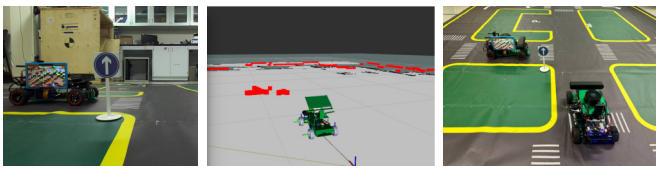
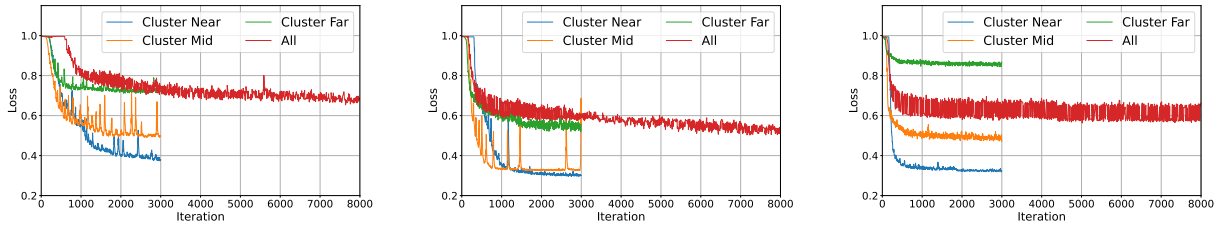


Fig. 6: Experimental setup. Left column: images recorded by the `camera car`; Middle column: point clouds representing the *patch car* and the target sign within the map; Right column: the top view of the two cars.

Autonomous driving scenarios. We consider various robot positional settings at the intersections where the traffic signs are crucial for decision-making. When the `patch car` must yield to the `camera car` according to the traffic signs, it

[1]https://github.com/AmirhoseinCh/DynamicPatch



(a) Loss for the "Go-straight" sign attack (b) Loss for the "Turn" sign attack (c) Loss for the "Pedestrian" sign attack

Fig. 7: Patch training loss for different traffic sign attacks, showing the rapid convergence of training for individual clusters compared to the entire dataset.

aims to alter the class of the detected signs by the `camera car` to pass the intersection earlier. In this setting, both cars approach an intersection within the same time window while driving on crossed roads, and the `patch car` tries to pass the intersection before the `camera car`. There are three possible traffic signs: Go-straight, Turn, and Pedestrian crossing signs. The sign is targeted by the patch attack to be misclassified as a stop sign. Figure 6 shows our setup for the experiment.

B. Attack Procedure

Step 1. We collect images from the display screen to construct the SIT-Net as explained in Section IV-B.

Step 2. We run robot cars in each scenario to collect synchronized Lidar and image data. This dataset is clustered as explained in Section IV-A. The number of clusters hinges on the total distance traversed and the precision of positional estimations. For our experiments, we set the cluster count to three. We utilize this processed dataset to optimize a set of patches using Algorithm 1. We set the training epoch at 1000 for each cluster and start with a randomly generated initial patch. Concurrently, a static patch is trained across all clusters for an identical epoch count.

Step 3. After training, we evaluate the effectiveness of the generated patches by the attack success rate in an extensive number of experiments including a similar environment and background and in settings with a background that is not in the attack dataset. For each of the sign types, approximately 1000 image frames are evaluated. An attack is considered successful if the patch makes the object detector predict a stop sign with confidence greater than 0.5 in place of the original sign.

We expand our evaluation to include comparisons with methodologies similar to ours, particularly focusing on sticker-based attacks as explained in [34], which represent the closest parallel in the existing literature. Unlike previous studies where patches are directly attached to the target object, in our approach, the adversarial patch and the target object are positioned separately. This unique setup necessitated the optimization of our patches through Equation III-C coupled with the consideration of the non-printability score, which is elaborated in [34, 28].

Moreover, we conduct experiments in which no patch was displayed to have a benign case. Table I presents the results of these experiments.

Attack Method	Dynamic (%)	Static (%)	Printed (%)	Benign (%)
Go-straight sign	39.9	30.0	1.3	0.0
Turn sign	27.4	22.1	3.0	0.0
Pedestrian sign	18.0	15.2	2.0	0.0

TABLE I: Comparison of attack success rates for Dynamic, Static, and Printed Patches across different sign types in an intersection scenario. This table also shows the control group ("Benign") where no adversarial attack was present, indicating no false detections under benign conditions.

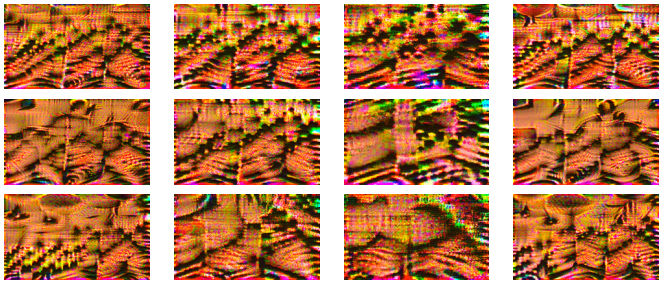
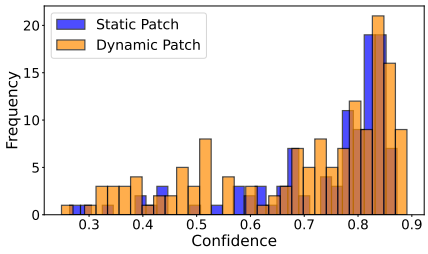


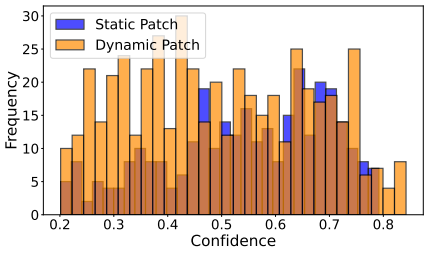
Fig. 8: Display of trained adversarial patches across various clusters and the entire dataset. The rows represent different traffic sign scenarios, specifically "Go-straight", "Turn", and "Pedestrian" signs, in that order. For the columns, from left to right, we showcase patches optimized for clusters based on proximity to the target: "Near", "Mid", and "Far", followed by patches trained on the combined dataset (denoted as 'All'). This arrangement illustrates how the patches are tailored to specific distances from the target object.

C. Results and Discussion

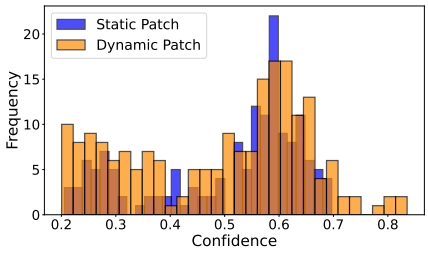
The loss over iterations during training of the patch for each cluster and all datasets is depicted in Figure 7. Also, Figure 8 shows the optimized patches for different clusters and the entire dataset. Figure 7 demonstrates that optimizing adversarial patches specifically for individual data clusters significantly expedites the training process and enhances the overall effectiveness of the attack. This cluster-based approach benefits from the inherent homogeneity within each cluster, allowing for more rapid optimization compared to a normal training approach on a broader dataset. The success rate in



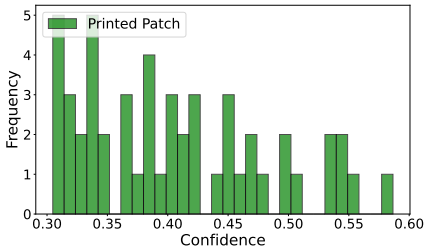
(a) Confidence distribution for misclassifying "Go-straight" sign as "Stop-sign"



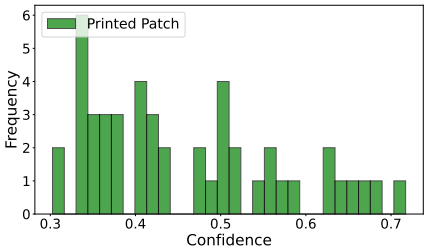
(b) Confidence distribution for misclassifying "Turn" sign as "Stop-sign"



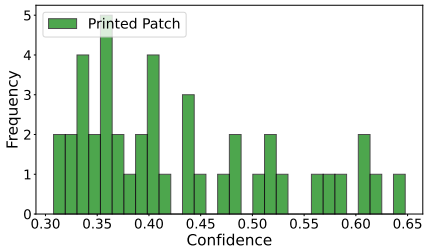
(c) Confidence distribution for misclassifying "Pedestrian" sign as "Stop-sign"



(d) Confidence distribution for misclassifying "Go-straight" sign as "Stop-sign" using a printed patch

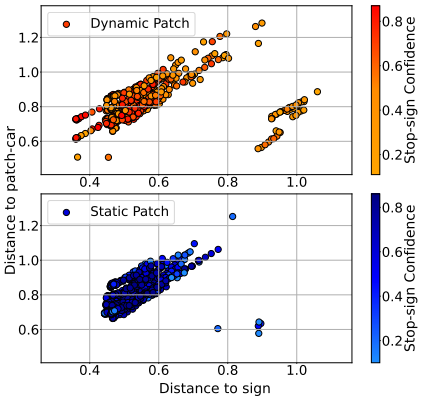


(e) Confidence distribution for misclassifying "Turn" sign as "Stop-sign" using a printed patch

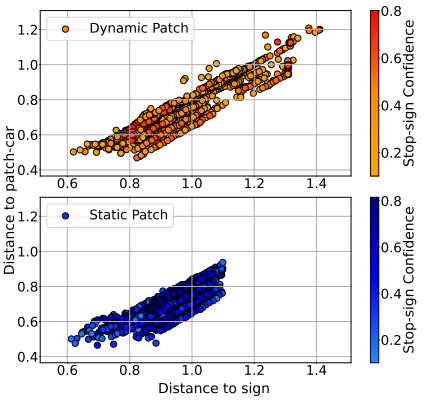


(f) Confidence distribution for misclassifying "Pedestrian" sign as "Stop-sign" using a printed patch

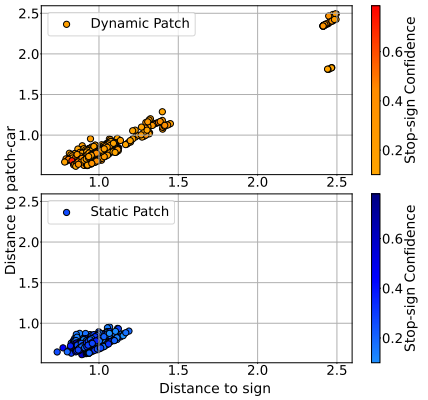
Fig. 9: Comparative histograms illustrating the confidence levels in misclassifying "Go-straight", "Turn", and "Pedestrian" signs as "Stop-sign" under two conditions: with dynamic/static patches (top row) and with printed adversarial patches (bottom row). These graphs highlight the effectiveness and variation of the adversarial attacks across different scenarios, noting that higher confidence values indicate better performance in the attacks. The spread and peaks of the confidence levels evidence the effectiveness of dynamic patches in deceiving.



(a) Effectiveness range of patches for the "Go-straight" sign attack



(b) Effectiveness range of patches for the "Turn" sign attack



(c) Effectiveness range of patches for the "Pedestrian" sign attack

Fig. 10: Scatter plots contrasting the effectiveness of dynamic and static patches in "Stop-sign" adversarial attacks as a function of the distance (in meters) of `camera car` to both the sign and the `patch car`. The dynamic patches (displayed in red-orange) demonstrate higher effectiveness across a more extensive range of distances from the camera to the patch, as indicated by the spread of the data points. The color intensity reflects the classifier's confidence in misclassification, with the dynamic patch showing high confidence over larger distances compared to the static patch (shown in blue).

Table I indicates that Dynamic patches consistently outperform Static and Printed patches that attempt to cover all clusters in the dataset. Printed patches show significantly lower effectiveness across all sign types. We discuss the ineffectiveness of these printed patches in Appendix C. The benign case, with a 0.0% success rate, confirms the absence of false positives in normal conditions. Further, Figure 9 displays the confidence distribution for the misclassification of "Stop-sign" for three sets of adversarial patch attacks on the "Pedestrian", "Turn", and "Go-straight". The dynamic patch consistently outperforms the static patch in every case in terms of confidence level. A visual depiction of the attack efficiency corresponding to the `camera car`'s distance from the `patch car` and the sign is shown in Figure10. In contrast to the static patch, which has a more concentrated cluster of effectiveness at shorter ranges, the dynamic patch's effectiveness does not decrease as significantly with greater distance. Overall, Figure 9 and Figure 10 point to the fact that dynamic patches are more successful than static patches in deceiving the object detector over a variety of instances. This can be explained by the adaptive nature of dynamic patches, which are designed to take into account different perspectives and distances, making them more resilient to environmental changes. This is visually represented in Figure 12, which depicts the consecutive frames of the attacks. A video showing attacks using dynamic patches in various autonomous driving scenarios involving two R2 robot cars is attached. [2]

Moreover, the designed objective function demonstrates the efficiency in generating a stop sign in the location of the original traffic sign as shown in Figure 12. This functionality is essential for the success of the adversarial attack, as it directly influences the decision-making algorithms of the autonomous driving systems being attacked. By incorrectly identifying key traffic signs, the dynamic patches deceive the autonomous vehicle into stopping or passing prematurely, leading to incorrect navigational choices.

D. Ablation Study on the Effectiveness of SIT-Net

To evaluate the necessity and impact of the Screen Image Transformation Network (SIT-Net) within our adversarial framework, we conducted an ablation study where both static and dynamic adversarial patches were trained without incorporating SIT-Net. This experimental setup allowed us to isolate and study the specific contributions of SIT-Net, which simulates realistic environmental influences like color transformation and contrast adjustments that are crucial in real-world scenarios.

In our simulations, we overlaid the trained patches on images to mimic their real-world deployment on screens. The patches performed effectively under these ideal conditions, demonstrating their potential efficacy. However, when these patches were actually displayed on screens in a real-world environment, their effectiveness drastically declined, with a success rate dropping to zero. This significant discrepancy underscores the critical role of SIT-Net in preparing adversarial patches to cope with real-world variability and environmental conditions.

Furthermore, we found that simulations lacking SIT-Net do not adequately capture the visual characteristics of adversarial patches as displayed in real-world conditions. In simulations, the patches may appear sharper and have clearer colors than they do in real-world environments. This difference results in an overestimation of patch effectiveness during optimization, as the simulated visual quality exceeds what is typically achievable in practical deployments. This emphasizes the essential role of SIT-Net in providing a realistic preparation and evaluation of adversarial patches, ensuring that their performance under real-world conditions is accurately represented.

Figure 11 showcases the dramatic changes in patches when displayed on a screen, highlighting the importance of including SIT-Net in the training process to achieve realistic and effective adversarial patches in practical deployments.

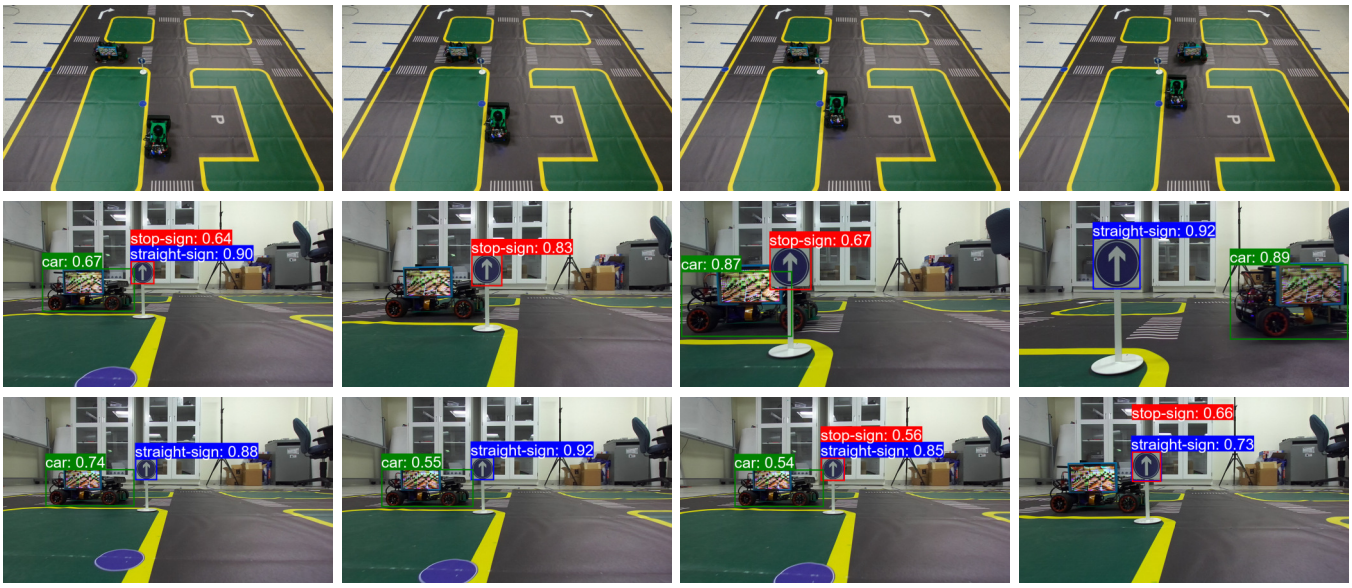
Fig. 11: Illustration of adversarial patch effectiveness without SIT-Net. The top row depicts real-world implementation results, and the bottom row shows simulation results. The columns, from left to right, represent attacks on "Go-straight", "Turn", and "Pedestrian" signs. This visual comparison clearly shows that the simulation without SIT-Net fails to replicate the real-world effectiveness of the patches, emphasizing the importance of SIT-Net in realistic adversarial training.

E. Conclusions

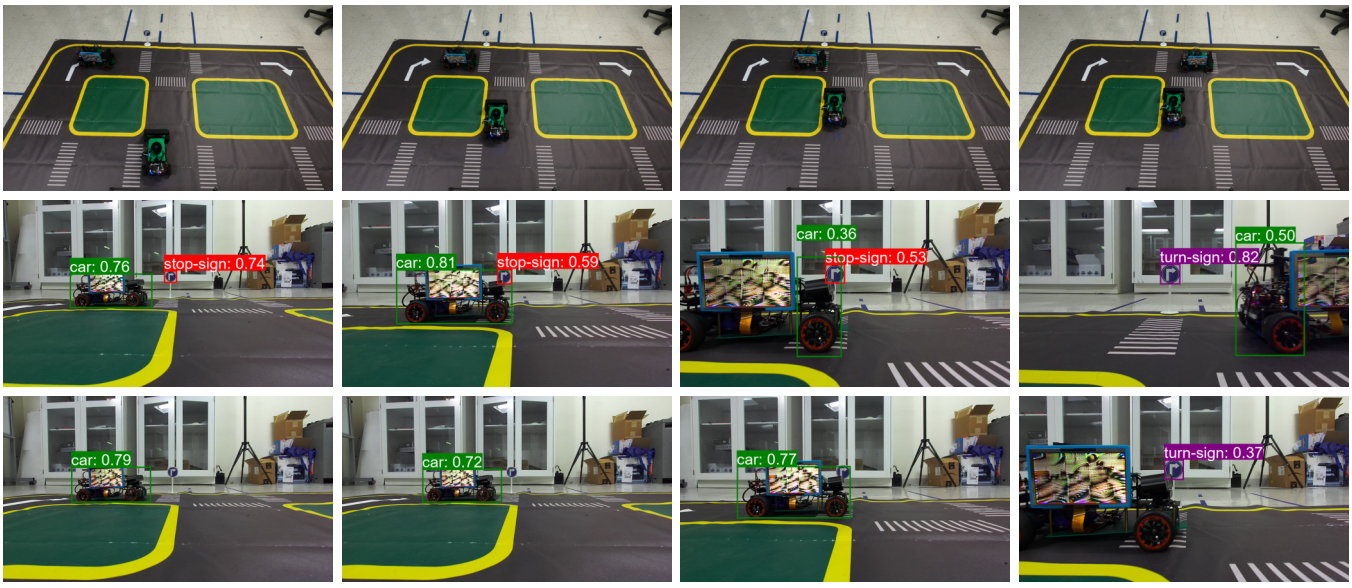
We formulate a novel approach for attacking object detectors by employing dynamic patches displayed on a screen to influence the decisions of autonomous driving systems. We have demonstrated that data clustering substantially accelerates the training process and enhances the overall efficacy of the dynamic attack. Additionally, to address the issue of color transformation in captured images of the displayed patch, we have introduced a straightforward CNN model capable of compensating for environmental factors.

Our findings underscore the potential vulnerabilities in the perception mechanisms of autonomous driving systems and highlight the importance of considering such adversarial tactics in the development of robust decision-making algorithms. Future work will focus on developing strategies to enhance the robustness of these systems against adversarial attacks. This includes exploring advanced machine learning techniques to detect and mitigate the effects of dynamic adversarial patches in real-time, as well as enhancing the training methods to incorporate a wider range of environmental variations.

[2]https://youtu.be/Wh2sPYpWczQ



(a) Adversarial patch attacks on the Go-straight sign.



(b) Adversarial patch attacks on the Turn sign.

Fig. 12: Sequential frames depicting two types of adversarial patch attacks on an autonomous vehicle's perception system. For each sub-figure, the top row shows the top view of a dynamic patch attack scenario. The middle row illustrates a dynamic patch attack, wherein the adversarial patches displayed on the screen change based on the distances to the camera and the sign, leading to a more deceptive misclassification across consecutive frames. The bottom row presents a static patch attack, where a single, unchanging patch is used. Both attacks aim to disrupt the perception system, causing it to mistakenly identify the signs as Stop signs.

References

[1] Alireza Asvadi, Luis Garrote, Cristiano Premebida, Paulo Peixoto, and Urbano J. Nunes. Multimodal vehicle detection: fusing 3d-lidar and color camera data. *Pattern Recognition Letters*, 115:20–29, 2018. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2017.09.038. URL https://www.sciencedirect.com/science/article/pii/S0167865517303598. Multimodal Fusion for Pattern Recognition.

[2] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. *CoRR*, abs/1707.07397, 2017. URL http://arxiv.org/abs/1707.07397.

[3] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *ArXiv*, abs/1712.09665, 2017. URL https://api.semanticscholar.org/CorpusID:13198654.

[4] Yulong Cao, Chaowei Xiao, Dawei Yang, Jing Fang, Ruigang Yang, Mingyan Liu, and Bo Li. Adversarial objects against lidar-based autonomous driving systems. *CoRR*, abs/1907.05418, 2019. URL http://arxiv.org/abs/1907.05418.

[5] Kejiang Chen, Yuefeng Chen, Hang Zhou, Chuan Qin, Xiaofeng Mao, Weiming Zhang, and Nenghai Yu. Adversarial examples detection beyond image space. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3850–3854, 2021. doi: 10.1109/ICASSP39728.2021.9414008.

[6] Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng Chau. Robust physical adversarial attack on faster R-CNN object detector. *CoRR*, abs/1804.05810, 2018. URL http://arxiv.org/abs/1804.05810.

[7] Andrew Du, Bo Chen, Tat-Jun Chin, Yee Wei Law, Michele Sasdelli, Ramesh Rajasegaran, and Dillon Campbell. Physical adversarial attacks on an aerial imagery object detector. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1796–1806, January 2022.

[8] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: from adversarial to random noise. *CoRR*, abs/1608.08967, 2016. URL http://arxiv.org/abs/1608.08967.

[9] Keke Geng, Ge Dong, Guodong Yin, and Jingyu Hu. Deep dual-modal traffic objects instance segmentation method using camera and lidar data for autonomous driving. *Remote Sensing*, 12(20):3274, October 2020. ISSN 2072-4292. doi: 10.3390/rs12203274. URL http://dx.doi.org/10.3390/rs12203274.

[10] Brian P. Gerkey. amcl: Adaptive monte carlo localization. ROS Wiki, Accessed: 2023. Available: http://wiki.ros.org/amcl.

[11] Jacob Gildenblat and contributors. Pytorch library for cam methods. https://github.com/jacobgil/pytorch-grad-cam, 2021.

[12] Shahar Hoory, Tzvika Shapira, Asaf Shabtai, and Yuval Elovici. Dynamic adversarial patch for evading object detection models. *CoRR*, abs/2010.13070, 2020. URL https://arxiv.org/abs/2010.13070.

[13] Lifeng Huang, Chengying Gao, Yuyin Zhou, Changqing Zou, Cihang Xie, Alan L. Yuille, and Ning Liu. UPC: learning universal physical camouflage attacks on object detectors. *CoRR*, abs/1909.04326, 2019. URL http://arxiv.org/abs/1909.04326.

[14] Wei Jia, Zhaojun Lu, Haichun Zhang, Zhenglin Liu, Jie Wang, and Gang Qu. Fooling the eyes of autonomous vehicles: Robust physical adversarial examples against traffic sign recognition systems. Internet Society, 4 2022. doi: 10.14722/ndss.2022.24130.

[15] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, ChristopherSTAN, Liu Changyu, Laughing, tkianai, Adam Hogan, lorenzomammana, yxNONG, AlexWang1900, Laurentiu Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, Francisco Ingham, Frederik, Guilhen, Hatovix, Jake Poznanski, Jiacong Fang, Lijun Yu, changyu98, Mingyu Wang, Naman Gupta, Osama Akhtar, PetrDvoracek, and Prashant Rai. ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements, October 2020. URL https://doi.org/10.5281/zenodo.4154370.

[16] Mark Lee and Zico Kolter. On physical adversarial patches for object detection. 6 2019. URL http://arxiv.org/abs/1906.11897.

[17] Juncheng Li, Frank Schmidt, and Zico Kolter. Adversarial camera stickers: A physical camera-based attack on deep learning systems. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3896–3904. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/li19j.html.

[18] Giulio Lovisotto, H.C.M. Turner, Ivo Sluganovic, Martin Strohmeier, and Ivan Martinovic. Slap: Improving physical adversarial examples with short-lived adversarial perturbations. In *USENIX Security Symposium*, 2020. URL https://api.semanticscholar.org/CorpusID:220403405.

[19] Jiajun Lu, Theerasit Issaranon, and David A. Forsyth. Safetynet: Detecting and rejecting adversarial examples robustly. *CoRR*, abs/1704.00103, 2017. URL http://arxiv.org/abs/1704.00103.

[20] Jiajun Lu, Hussein Sibai, Evan Fabry, and David A. Forsyth. NO need to worry about adversarial examples in object detection in autonomous vehicles. *CoRR*, abs/1707.03501, 2017. URL http://arxiv.org/abs/1707.03501.

[21] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5188–5196, 2015. doi: 10.1109/CVPR.2015.7299155.

[22] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi,

Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. *CoRR*, abs/1610.08401, 2016. URL http://arxiv.org/abs/1610.08401.

[23] Mohammed Bany Muhammad and Mohammed Yeasin. Eigen-cam: Class activation map using principal components. In *2020 international joint conference on neural networks (IJCNN)*, pages 1–7. IEEE, 2020.

[24] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *CoRR*, abs/1412.1897, 2014. URL http://arxiv.org/abs/1412.1897.

[25] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[26] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL http://arxiv.org/abs/1506.01497.

[27] Sara Sabour, Yanshuai Cao, Fartash Faghri, and David J. Fleet. Adversarial manipulation of deep representations, 2015. URL https://arxiv.org/abs/1511.05122.

[28] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 1528–1540, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341394. doi: 10.1145/2976749.2978392. URL https://doi.org/10.1145/2976749.2978392.

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. 9 2014. URL http://arxiv.org/abs/1409.1556.

[30] Dawn Song, Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramèr, Atul Prakash, and Tadayoshi Kohno. Physical adversarial examples for object detectors. In *12th USENIX Workshop on Offensive Technologies (WOOT 18)*, Baltimore, MD, August 2018. USENIX Association. URL https://www.usenix.org/conference/woot18/presentation/eykholt.

[31] Stereolabs. Zed 2 - ai stereo camera, 2023. URL https://www.stereolabs.com/zed-2/. Technical specifications and features of ZED 2 camera.

[32] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/f7cade80b7cc92b991cf4d2806d6bd78-Paper.pdf.

[33] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2013. URL https://arxiv.org/abs/1312.6199.

[34] Simen Thys, Wiebe Van Ranst, and Toon Goedemé. Fooling automated surveillance cameras: adversarial patches to attack person detection. *CoRR*, abs/1904.08653, 2019. URL http://arxiv.org/abs/1904.08653.

[35] Jessica Van Brummelen, Marie O'Brien, Dominique Gruyer, and Homayoun Najjaran. Autonomous vehicle perception: The technology of today and tomorrow. *Transportation Research Part C: Emerging Technologies*, 89:384–406, 2018. ISSN 0968-090X. doi: https://doi.org/10.1016/j.trc.2018.02.012. URL https://www.sciencedirect.com/science/article/pii/S0968090X18302134.

[36] Jiakai Wang, Aishan Liu, Zixin Yin, Shunchang Liu, Shiyu Tang, and Xianglong Liu. Dual attention suppression attack: Generate adversarial camouflage in physical world. *CoRR*, abs/2103.01050, 2021. URL https://arxiv.org/abs/2103.01050.

[37] Ningfei Wang, Yunpeng Luo, Takami Sato, Kaidi Xu, and Qi Alfred Chen. Does physical adversarial example really matter to autonomous driving? towards system-level effect of adversarial object evasion attack. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4412–4423, 2023.

[38] Liang Xie, Chao Xiang, Zhengxu Yu, Guodong Xu, Zheng Yang, Deng Cai, and Xiaofei He. Pi-rcnn: An efficient multi-sensor 3d object detector with point-based attentive cont-conv fusion module. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07): 12460–12467, April 2020. ISSN 2159-5399. doi: 10.1609/aaai.v34i07.6933. URL http://dx.doi.org/10.1609/aaai.v34i07.6933.

[39] Kaidi Xu, Sijia Liu, Pu Zhao, Pin-Yu Chen, Huan Zhang, Quanfu Fan, Deniz Erdogmus, Yanzhi Wang, and Xue Lin. Structured adversarial attack: Towards general implementation and better interpretability. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BkgzniCqY7.

[40] Kaidi Xu, Gaoyuan Zhang, Sijia Liu, Quanfu Fan, Mengshu Sun, Hongge Chen, Pin-Yu Chen, Yanzhi Wang, and Xue Lin. Adversarial t-shirt! evading person detectors in a physical world. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 665–681, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58558-7.

[41] Yahboom Technology. Rosmaster r2 study material. Yahboom Technology, Accessed: 2023. Available: http://www.yahboom.net/study/ROSMASTER-R2.

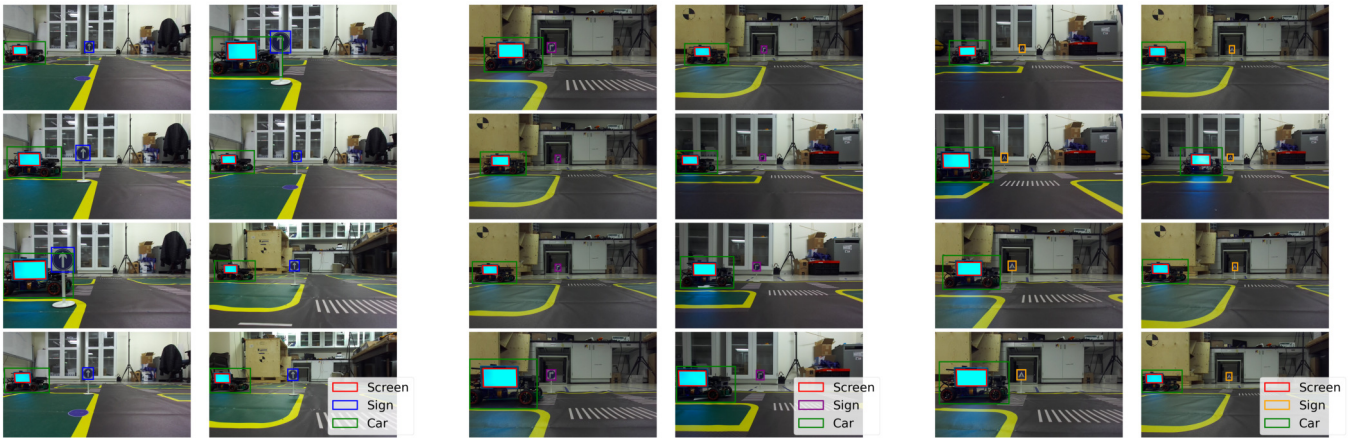
[42] Pu Zhao, Kaidi Xu, Sijia Liu, Yanzhi Wang, and Xue Lin. Admm attack: an enhanced adversarial attack for deep neural networks with undetectable distortions. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, pages 499–505, 2019.

[43] 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 *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 1989–2004, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367479. doi: 10.1145/3319535.3354259. URL https://doi.org/10.1145/3319535.3354259.

APPENDIX

A. *Dataset Overview*

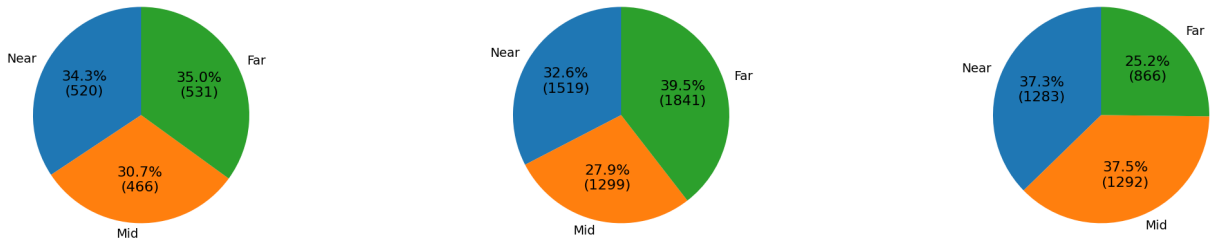
The dataset utilized for training our adversarial patches was meticulously compiled to ensure a broad representation of various scenarios encountered in autonomous driving. This dataset includes synchronized LiDAR and image data collected from our robot cars in indoor environments for each sign, which is critical for training both static and dynamic adversarial patches effectively. Figure 13 displays examples from our dataset, showing the diversity in the images. The full dataset is provided in this link[3].



(a) Go-straight sign (b) Turn sign (c) Pedestrian sign

Fig. 13: Example images from dataset for each sign, demonstrating the variety of scenarios in the dataset.

The dataset for each sign is divided into three distinct clusters based on the distance to the screen and sign as discussed in section IV-A2, which are pivotal for the effectiveness of dynamic patches in varying scenarios. These clusters are specifically designed to encapsulate different ranges of distances between the camera car and the target objects, ensuring that each patch is optimized for its respective range. The clustering approach helps in tailoring the patches to specific environmental conditions and object proximities. Figure 14 provides a detailed breakdown of the number of images in each cluster for each type of sign.



(a) Go-straight sign (b) Turn sign (c) Pedestrian sign

Fig. 14: Distribution of images across the three clusters, showing the balance of clusters for each sign.

[3]https://drive.google.com/drive/folders/1UiODhj44Wos0TJAiK1067lCwvnoJt0qu?usp=sharing

B. Natural Lightening Condition Test

We conducted additional experiments to assess the robustness of our adversarial patches under various outdoor lighting conditions using patches that were initially trained in an indoor environment. Even though our adversarial patches were trained for indoor controlled conditions, they remained effective in outdoor lighting conditions. Figure 15 illustrates the performance of adversarial patches under different natural lighting conditions at various times of the day. These results highlight the resilience of our approach across a broad spectrum of natural lighting situations, though they also highlight areas for potential enhancement, particularly in very harsh lighting conditions.

Fig. 15: Adversarial patch attacks in natural lighting conditions at different times of the day.

Additionally, to quantify the effectiveness more precisely, we generated scatter plots contrasting the performance of dynamic and static patches. Figure 16 provides a detailed view of the "Stop-sign" adversarial attacks and their effectiveness as a function of the distance (in meters) between the camera car and both the sign and the patch car in outdoor lighting conditions.

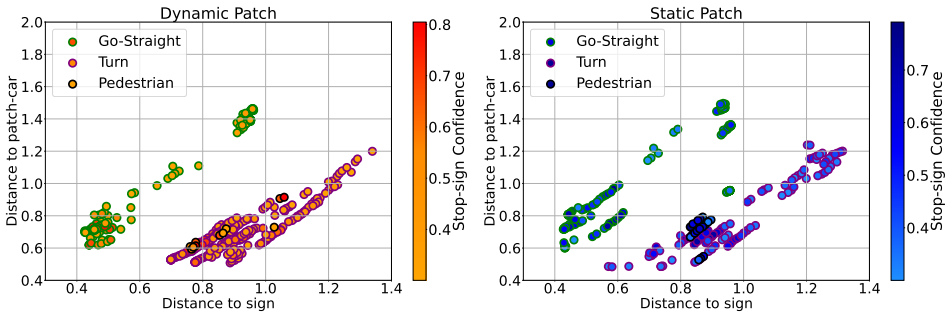


Fig. 16: Scatter plots contrasting the effectiveness of dynamic and static patches in "Stop-sign" adversarial attacks as a function of the distance (in meters) of `camera car` to both the sign and the `patch car` in the outdoor lighting condition.

C. *Ineffectiveness of Printed Patches in Real-World Scenarios*

We present the extended analysis of printed adversarial patches, highlighting the practical implications of the non-printability loss on real-world efficacy. The non-printability score, as discussed in [34], ensures that the colors utilized in the patches can be accurately reproduced by standard printers. However, this requirement often limits the overall effectiveness of the attack by restricting the range of colors that can be used. This constraint leads to a significant reduction in the effectiveness of the printed patches.

Figure 17 illustrates that the confidence levels for the "stop-sign" induced by these patches consistently fall below the critical threshold of 0.5. This outcome directly reflects the limitations imposed by the non-printability loss, which, while intended to make the patches printable, does not adequately mirror the appearance of real-world adversarial patches.

Fig. 17: Sequential frames showing the performance of adversarial printed patch attacks. The frames depict that the confidence levels of **the misclassified "Stop-sign" remain below the critical threshold of 0.5**, indicating the ineffectiveness of these patches in real-world conditions.