A. Adversarial attacks algorithms

We present algorithms for both our PGD (Algorithm 1) and universal (Algorithm 2) attacks. In both cases we make use of the PGD adversarial attack scheme [5] to optimize a single adversarial patch. In each optimization step, we update the patch based on the gradient of the training criterion. Finally, we return the produced patch which maximized the evaluation criterion.

Algorithm 1 PGD adversarial attack

Input VO: VO model
Input A: Adversarial patch perturbation
Input (x, y): Trajectory to attack and it's ground truth motions
Input $(\ell_{train}, \ell_{eval})$: Train and evaluation loss functions
Input α: Step size for the attack

$P \leftarrow \text{Uniform}(0, 1)$
$P_{\text{best}} \leftarrow P$
$\text{Loss}_{\text{best}} \leftarrow 0$
for $k = 1$ to K **do**
  optimization step:
  $g \leftarrow \nabla_P \ell_{train}(VO(A(x, P)), y)$
  $P \leftarrow P + \alpha \cdot \text{sign}(g)$
  $P \leftarrow clip(P, 0, 1)$
  evaluate patch:
  $\text{Loss} \leftarrow \ell_{eval}(VO(A(x, P)), y)$
  if $\text{Loss} > \text{Loss}_{\text{best}}$ **then**
    $P_{\text{best}} \leftarrow P$
    $\text{Loss}_{\text{best}} \leftarrow \text{Loss}$
  end if
end for
return P_{best}

B. Experimental setting and VO model

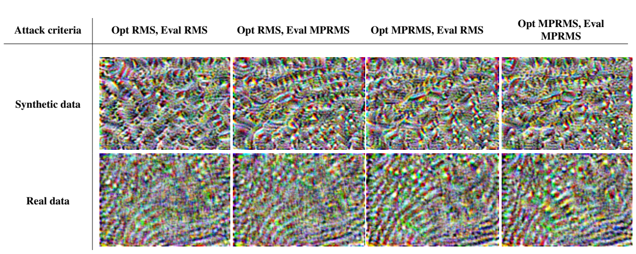


Figure 6. Visualization of universal adversarial patches. For each dataset, and optimization and evaluation criteria, we present the universal adversarial image produced via the in-sample attack scheme.

Algorithm 2 Universal PGD adversarial attack

Input VO: VO model
Input A: Adversarial patch perturbation
Input (X_{train}, Y_{train}): Trajectories training dataset
Input (X_{eval}, Y_{eval}): Trajectories evaluation dataset
Input $(\ell_{train}, \ell_{eval})$: Training and evaluation loss functions
Input (N_{train}, N_{eval}): Number of training and evaluation trajectories
Input α: Step size for the attack

$P \leftarrow \text{Uniform}(0, 1)$
$P_{\text{best}} \leftarrow P$
$\text{Loss}_{\text{best}} \leftarrow 0$
for $k = 1$ to K **do**
  optimization step:
  $g \leftarrow 0$
  for $i = 1$ to N_{train} **do**
    $\hat{y}_{train,i} \leftarrow VO(A(x_{train,i}, P))$
    $g \leftarrow g + \nabla_P \ell_{train}(\hat{y}_{train,i}, y_{train,i})$
  end for
  $P \leftarrow P + \alpha \cdot \text{sign}(g)$
  $P \leftarrow clip(P, 0, 1)$
  evaluate patch:
  $\text{Loss} \leftarrow 0$
  for $i = 1$ to N_{eval} **do**
    $\hat{y}_{eval,i} \leftarrow VO(A(x_{eval,i}, P))$
    $\text{Loss} \leftarrow \text{Loss} + \ell_{eval}(\hat{y}_{eval,i}, y_{eval,i})$
  end for
  if $\text{Loss} > \text{Loss}_{\text{best}}$ **then**
    $P_{\text{best}} \leftarrow P$
    $\text{Loss}_{\text{best}} \leftarrow \text{Loss}$
  end if
end for
return P_{best}

B.1. Experimental setting

In this section we further detail the distinct experimental settings of in-sample, out-of-sample and closed-loop.

B.1.1 In-Sample

The in-sample setting is used to estimate the effect of universal and PGD adversarial perturbations on known data. We train, evaluate and test our attack on the entire dataset. We then compare the best performing attack to the random and clean baselines, for both our PGD and universal adversarial attacks.

B.1.2 Out-Of-Sample

The out-of-sample setting is used to estimate generalization properties of universal perturbations to unseen data. Our methodology in this setting is to first split the trajectories into several folders, each with distinct initial positions of the contained trajectories. Thereafter, we perform cross-validation over the folders, where in each iteration a distinct folder is chosen to be the test set, and another to be the evaluation set. The training set thus comprises of the remaining folders. We report the average results over the test sets. Throughout our experiments, we use 10-fold cross validation.

B.1.3 Closed-Loop

The closed-loop setting is used to estimate the generalization properties of the previously produced adversarial patches to a closed-loop scheme, in which the outputs of the VO model are used in a simple navigation scheme. Our navigation scheme is an aerial path follower based on the carrot chasing algorithm [9]. Given the current pose, target position and cruising speed, the algorithm computes a desired motion toward the target position. We then produce trajectories, each with a distinct initial and target position, with the motions computed iteratively by the navigation scheme based on the provided current position. The ground truth trajectories are computed by providing the current position in each step as the aggregation of motions computed by the navigation scheme. The estimated trajectories for a given patch, clean or adversarial, however, are computed by providing the current position in each step as the aggregation of motions estimated by the VO, where the viewpoint in each step corresponds to the aggregation of motions computed by the navigation scheme. We chose this navigation scheme to further assess the incremental effect of our adversarial attacks, as any deviation in the VO estimations directly affects the produced trajectory.

B.2. VO model

The VO model used in our experiments is the TartanVO [11], a recent differentiable VO model that achieved state-of-the-art performance in visual odometry benchmarks. Moreover, to better generalize to real-world scenarios, the model was trained over scale-normalized trajectories in diverse synthetic datasets. As the robustness of the model improves on scale-normalized trajectories, we supply it with the scale of the ground truth motions. The assumption of being aware of the motions' scale is a reasonable one, as the scale can be estimated to a reasonable degree in typical autonomous systems from the velocity. In our experiments, we found that the model yielded plausible trajectory estimates over the clean trajectories, for both synthetic and real data.

C. Data generation specifics

In this section we detail and provide specifics for the generation process of both the synthetic and real datasets.

C.1. Synthetic data

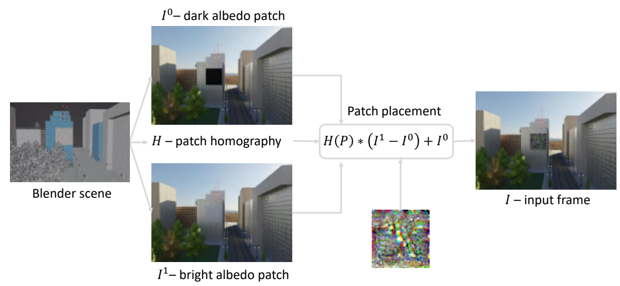


Figure 7. Synthetic frame generation. The attack patch P is projected via the homography transformation H and is incorporated into the scene according to the albedo images I_0 and I_1.

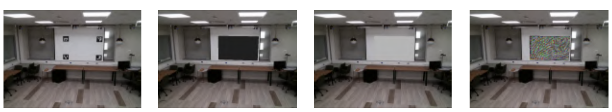
As mentioned in Sec. 3, The renderer framework used for the syntetic data is Blender [1]. To accurately estimate the motions using the VO model, we require photo-realistic rendering. In addition, the whole scene is altered for each camera motion, mandating re-rendering for each frame. Online rendering is impractical for optimization, in terms of computational overhead. Offline rendering is sufficient for our optimization schemes, and only our closed-loop test requires online rendering. We, therefore, produce the data for optimizing the adversarial patches offline, and make use of online rendering only for the closed-loop test.

In the offline data generation of each trajectory $\{I_t\}_{t=0}^{L}$, we produce $\{I_t^0\}_{t=0}^{L}, \{I_t^1\}_{t=0}^{L}, \{H_t\}_{t=0}^{L}$, as well as the ground truth camera motions δ_t^{t+1}. For the closed-loop test, for each initial position, target position and pre-computed patch P, we compute the ground truth motions δ_t^{t+1} and their estimation by the VO model, as described in Appendix B.1.3.

Offline rendered data specifics We produced 100 trajectories with a constant linear velocity norm of $v = 5[\frac{m}{s}]$, and a constant $2D$ angular velocity sampled from $v_\theta = \mathcal{N}(0,3)[\frac{deg}{s}]$. Each trajectory is nearly $10[m]$ long and contains 60 frames at 30 fps. The trajectories are evenly divided between 10 initial positions, with the initial positions being distributed evenly on the ring of a right circular cone with a semi-vertical angle of $10°$ and a $50[m]$-long axis aligned with the patch's normal. We used a camera with a horizontal field-of-view (FOV) of $80°$ and 640×448 resolution. The patch is a $30[m]$ square, occupying, under the above conditions, an average FOV over the trajectories ranging from 18.1% to 27.3%, and covering a mean 22.2% of the images. To estimate the effect of the patch's size, which translates into a ℓ_0 limitation on the adversarial attacks, we also make use of smaller sized patches. The outer margins of the patch would then be defaulted to the clean I_0 image, and the adversarial image would be projected onto a smaller sized square, with its center aligned as before.

Closed-loop data specifics Similarly, in the closed-loop scheme we produced trajectories with the same camera and patch configuration, the same distribution of initial positions and with the navigation scheme cruising speed set according to the previous linear velocity norm of $v = 5[\frac{m}{s}]$. Here we, however, produce 10 trajectories by randomly selecting a target position at the proximity of the patch for each initial position. We then produced the ground truth and VO trajectories for each patch P as described in Appendix B.1.3. The trajectories are each $45[m]$ long and contain 270 frames at 30 fps.

C.2. Real data



(a) (b) (c) (d)

Figure 8. Real dataset frame generation. (a) Original image. (b+c) Black and white albedo approximations. (d) Adversarial patch projected onto the scene.

Similarly to the offline synthetic data generation, for each trajectory $\{I_t\}_{t=0}^{L}$ we produced $\{I_t^0\}_{t=0}^{L}, \{I_t^1\}_{t=0}^{L}, \{H_t\}_{t=0}^{L}$ as well as the ground truth camera motions δ_t^{t+1}.

We produced 48 trajectories with a constant velocity norm of approximately $v \simeq 1[\frac{m}{s}]$. Each trajectory contained 45 frames at 30 fps with total length $l \sim \mathcal{N}(1.56, 0.15^2)[m]$. The trajectories' initial positions were distributed evenly on a plane parallel to the patch at a distance of $7.2[m]$. Not including the drone movement model, the trajectories comprised linear translation toward evenly distributed target positions at the patch's plane. We used a camera with a horizontal FOV of $82.6°$, and 640×448 resolution. The patch was a $1.92 \times 1.24[m]$ rectangle, occupying, under the above conditions, an average FOV over the trajectories ranging from 6.8% to 11.2%, and covering a mean 8.8% of the images.

D. Additional experiments

In this section we present additional experimental results.

In Fig. 9 we show the patch size comparison of the

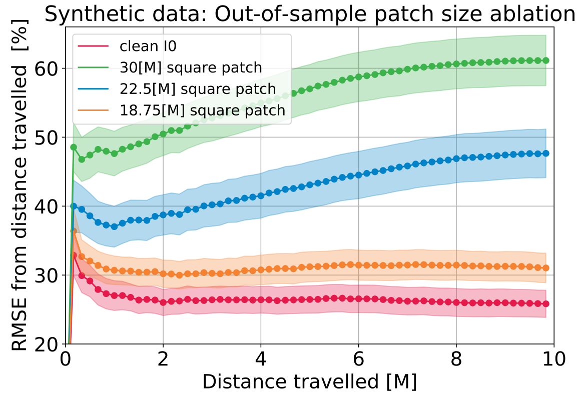


Figure 9. A comparison of different patch sizes of the accumulated deviation in distance travelled from the ground-truth trajectories over out-of-sample cross-validation of the synthetic dataset as a function of the trajectory length. The patches are a $30[m]$, a $22.5[m]$, and $18.75[m]$ squares and occupy a FOV over the trajectories ranging from $18.1\% - 27.4\%, 8.3\% - 12.6\%, 6.1\% - 9.3\%$ respectively, and covering a mean $22.2\%, 10.2\%, 7.5\%$ of the images. We show a comparison of the deviation in distance travelled between our best performing universal attacks for each patch and the clean baseline.

out-of-sample results on the synthetic dataset. Our universal attacks again showed a substantial increase in the generated deviation over the clean baseline, however, as the patch size is reduced, the increase in the generated deviation becomes less significant. For the $30[m]$ square patch, the best performing universal attack generated, after $10[m]$, a deviation of 61% in distance travelled. For the same configuration, the best performing universal attack for the $22.5[m]$ square patch generated a deviation of 48% in distance travelled. Regarding the $18.75[m]$ square patch, the generated deviation decays significantly with the best performing universal attack generating, after $10[m]$, a deviation of 31% in distance travelled.