Project guideline: Universal adversarial perturbations on visual odometry systems

1 Inroduction

Deep neural networks are known to be susceptible to adversarial perturbations, small perturbation which alter the output of the network and exist under strict norm limitations. While such perturbations are usually tailored to a specific input, it is also possible to produce universal perturbations that achieve a high rate of miss-classification on a set of inputs. Universal perturbations present a more realistic use case for adversarial perturbations, as the adversary does not require to be aware of the model's input. In this project you will research the susceptibility of visual odometry (VO) models to universal adversarial perturbations. A VO model aims to infer the motion (position and orientation) between two respective viewpoints. Such models are frequently used by visual based autonomous navigation systems. In this project you will produce universal adversarial perturbations for a given VO model, aiming to maximise its physical 3D deviation.

General Note This is a research project. Although much of the necessary guidance is provided below, you are expected to research the topic and read relevant papers as needed in order to come up with solutions on your own. You will be graded based on both the quality of your report, which should detail the research you have conducted, and the performance of the methods you have devised.

2 Adversarial attacks on VO

In what follows, we start with a definition of the adversarial attack setting, for both the universal and non-universal cases. We continue to describe the adversarial optimization scheme used for producing the perturbations and discuss the optimization of the universal attacks aiming to perturb unseen data.

2.1 Patch adversarial attack setting

Patch adversarial perturbation. Let $\mathcal{I} = (0,1)^{3\times w\times h}$ be an RGB image space, for some width w and height h. For an image $I \in \mathcal{I}$, inserting a patch $P \in \mathcal{I}$ onto a given plane in I would then be a perturbation $A : \mathcal{I} \times \mathcal{I} \to \mathcal{I}$, and we denote $I^P = A(I, P)$. In order to compute I^P we first denote the black and white albedo images of the patch P as viewed from I viewpoint, $I^0, I^1 \in \mathcal{I}$. In addition, we denote the homography of the patch P to I viewpoint, H. While the albedo images essentially describe the dependency of I^P on the lighting conditions and the material of the patch P, H is only dependent on the relative camera motion between I and the patch P viewpoints. We now define I^P as:

$$I^P = A(I, P) = H \cdot P * (I^1 - I^0) + I^0 \tag{1}$$

where $*$ denotes element-wise multiplication. For a set of images $\{I_t\}$, we similarly define the perturbed set as inserting a single patch P onto the same plane in each of the images:

$$\{I_t^P\} = A(\{I_t\}, P) = \{H_t \cdot P \cdot (I_t^1 - I_t^0) + I_t^0\} \tag{2}$$

Attacking visual odometry Let $VO : \mathcal{I} \times \mathcal{I} \to \mathbb{R}^3 \times so(3)$ be a monocular visual odometry model, i.e for a given pair of consecutive images $\{I_t, I_{t+1}\}$ it estimates the relative camera motion $\delta_t^{t+1} = (T_t, R_t)$, where $T_t \in \mathbb{R}^3$ is the $3D$ translation and $R_t \in so(3)$ is the $3D$ rotation. We define a trajectory as a set of consecutive images $\{I_t\}_{t=0}^{L}$, for some length L, and extend the definition of the monocular visual odometry to trajectories:

$$VO(\{I_t\}_{t=0}^{L}) = \{\hat{\delta}_t^{t+1}\}_{t=0}^{L-1} \tag{3}$$

where $\hat{\delta}_t^{t+1}$ denotes the estimation of δ_t^{t+1} by the VO model. given a trajectory $\{I_t\}_{t=0}^{L}$, with ground truth motions $\{\delta_t^{t+1}\}_{t=0}^{L-1}$ and a criterion over the trajectory motions ℓ, an adversarial patch perturbation $P_a \in \mathcal{I}$ aims to maximize:

$$P_a = \arg\max_{P \in \mathcal{I}} \ell(VO(A(\{I_t\}_{t=0}^{L}, P)), \{\delta_t^{t+1}\}_{t=0}^{L-1}) \tag{4}$$

where A is defined as in Eq. (2). In this formalism, the limitation of the adversarial perturbation is expressed in the albedo images I^0, I^1 and can be described as dependent on the patch material. Similarly, for a set of trajectories $\{\{I_{i,t}\}_{t=0}^{L_i}\}_{i=0}^{N-1}$, with corresponding ground truth motions $\{\{\delta_{i,t}^{t+1}\}_{t=0}^{L_i-1}\}_{i=0}^{N-1}$, a universal adversarial attack aims to maximize the sum of the criterion over the trajectories:

$$P_{ua} = \arg\max_{P \in \mathcal{I}} \sum_{i=0}^{N-1} \ell(VO(A(\{I_{i,t}\}_{t=0}^{L_i}, P)), \{\delta_{i,t}^{t+1}\}_{t=0}^{L_i-1}) \tag{5}$$

However, differently from the non-universal adversarial perturbations which are tailored to a specific input, we can discuss the generalization properties of universal adversarial perturbation to unseen data. In which case the provided trajectories used for the optimization of the perturbation would differ from the test trajectories.

Task criterion For the scope of this project, the target criterion used for adversarial attacks is the RMS (root mean squares) deviation in the 3D physical translation between the accumulated trajectory motions, as estimated by the VO, and the ground truth. We denote the accumulated motion as δ_0^L, and it exist that:

$$\delta_0^L = \delta_0^1 \cdot \delta_1^2 \cdots \delta_{L-1}^L = \prod_{t=0}^{L-1} \delta_t^{t+1} \tag{6}$$

where the multiplication of motions is defined as the matrix multiplication of the corresponding 4×4 matrix representation:

$$\delta_t^{t+1} = \begin{pmatrix} R_t & T_t \\ \mathbf{0} & 1 \end{pmatrix} \tag{7}$$

The target criterion is then formulated as:

$$\ell_{VO}(VO(A(\{I_t\}_{t=0}^{L}, P)), \{\delta_t^{t+1}\}_{t=0}^{L-1}) = ||T(\prod_{t=0}^{L-1} VO(I_t^P, I_{t+1}^P)) - T(\prod_{t=0}^{L-1} \delta_t^{t+1})||_2 \tag{8}$$

where we denote $T(\delta_0^L) = T((T_0^L, R_0^L)) = T_0^L$.

2.2 Optimization of adversarial patches

We optimize the adversarial patch P via PGD adversarial attack with ℓ_{inf} norm. We limit the values in P to be in $[0, 1]$, however we do not enforce any additional ϵ limitation, as such would be expressed in the albedo images. For universal attacks, we allow for different train and evaluation datasets for the attack in order to enable evaluation on unseen data. We provide algorithms for both our universal (Algorithm 2) and non-universal (Algorithm 1) attacks. Notice that the non-universal attack is a specific case of the universal in which both train and evaluation data-sets are comprised of the same single trajectory.

Algorithm 1 Non-universal PGD adversarial attack

```
    Input $VO$: VO model
    Input $A$: Adversarial patch perturbation
    Input $x$: trajectory to attack
    Input $y$: trajectory ground truth motions
    Input $\ell_{train}$: train loss function
    Input $\ell_{eval}$: evaluation loss function
    Input $\alpha$: step size for the attack

 1: $P \leftarrow \text{Uniform}(0,1)$
 2: $P_{\text{best}} \leftarrow P$
 3: $\text{Loss}_{\text{best}} \leftarrow 0$
 4: for $k = 1$ to $K$ do
 5:     optimization step:
 6:     $\hat{y} \leftarrow VO(A(x,P))$
 7:     $g \leftarrow \nabla_P \ell_{train}(\hat{y}, y)$
 8:     $P \leftarrow P + \alpha \cdot \text{sign}(g)$
 9:     $P \leftarrow clip(P, 0, 1)$
10:     evaluate patch:
11:     $\hat{y} \leftarrow VO(A(x,P))$
12:     $\text{Loss} \leftarrow \ell_{eval}(\hat{y}, y)$
13:     if $\text{Loss} > \text{Loss}_{\text{best}}$ then
14:         $P_{\text{best}} \leftarrow P$
15:         $\text{Loss}_{\text{best}} \leftarrow \text{Loss}$
16:     end if
17: end for
18: return $P_{\text{best}}$
```

Optimization and evaluation criteria For either optimization and evaluation of attacks we consider one of 2 criteria. The first option which we denote as ℓ_{RMS} is a smoother version of the target criterion ℓ_{VO}, in which we sum over partial trajectories with the same origin as the full trajectory:

$$\ell_{RMS}(VO(A(\{I_t\}_{t=0}^{L}, P)), \{\delta_t^{t+1}\}_{t=0}^{L-1}) = \sum_{l=1}^{L} \ell_{VO}(VO(A(\{I_t\}_{t=0}^{l}, P)), \{\delta_t^{t+1}\}_{t=0}^{l-1}) \quad (9)$$

$$= \sum_{l=1}^{L} ||T(\prod_{t=0}^{l-1} VO(I_t^P, I_{t+1}^P)) - T(\prod_{t=0}^{l-1} \delta_t^{t+1})||_2 \quad (10)$$

The second option which we denote as ℓ_{MPRMS}, i.e mean partial RMS, is to take into account all the partial trajectories, however we take the mean for each length of partial trajectories in order to keep the factoring between different lengths as in ℓ_{RMS}. ℓ_{MPRMS} is more suited to generalization of universal attacks to unseen data than in-sample optimization, as it takes into consideration partial trajectories which may not be relevant to the full trajectory:

Algorithm 2 Universal PGD adversarial attack

```
Input $VO$: VO model
Input $A$: Adversarial patch perturbation
Input $X_{train}$: Training trajectories
Input $Y_{train}$: Training trajectories ground truth motions
Input $X_{eval}$: eval trajectories
Input $Y_{eval}$: eval trajectories ground truth motions
Input $\ell_{train}$: train loss function
Input $\ell_{eval}$: evaluation loss function
Input $N_{train}$: Number of train trajectories
Input $N_{eval}$: Number of eval trajectories
Input $\alpha$: step size for the attack

1: $P \leftarrow \text{Uniform}(0, 1)$
2: $P_{\text{best}} \leftarrow P$
3: $\text{Loss}_{\text{best}} \leftarrow 0$
4: for $k = 1$ to $K$ do
5:     optimization step:
6:     $g \leftarrow 0$
7:     for $i = 1$ to $N_{train}$ do
8:         $\hat{y}_{train,i} \leftarrow VO(A(x_{train,i}, P))$
9:         $g \leftarrow g + \nabla_P \ell_{train}(\hat{y}_{train,i}, y_{train,i})$
10:    end for
11:    $P \leftarrow P + \alpha \cdot \text{sign}(g)$
12:    $P \leftarrow clip(P, 0, 1)$
13:    evaluate patch:
14:    $\text{Loss} \leftarrow 0$
15:    for $i = 1$ to $N_{eval}$ do
16:        $\hat{y}_{eval,i} \leftarrow VO(A(x_{eval,i}, P))$
17:        $\text{Loss} \leftarrow \text{Loss} + \ell_{eval}(\hat{y}_{eval,i}, y_{eval,i})$
18:    end for
19:    if $\text{Loss} > \text{Loss}_{\text{best}}$ then
20:        $P_{\text{best}} \leftarrow P$
21:        $\text{Loss}_{\text{best}} \leftarrow \text{Loss}$
22:    end if
23: end for
24: return $P_{\text{best}}$
```

$$\begin{aligned}
&\ell_{MPRMS}(VO(A(\{I_t\}_{t=0}^{L}, P)), \{\delta_t^{t+1}\}_{t=0}^{L-1}) \\
&\quad = \sum_{l=1}^{L} \frac{1}{L-l+1} \sum_{i=0}^{L-l} \ell_{VO}(VO(A(\{I_t\}_{t=i}^{i+l}, P)), \{\delta_t^{t+1}\}_{t=i}^{i+l-1}) && (11) \\
&\quad = \sum_{l=1}^{L} \frac{1}{L-l+1} \sum_{i=0}^{L-l} ||T(\prod_{t=i}^{i+l-1} VO(I_t^P, I_{t+1}^P)) - T(\prod_{t=i}^{i+l-1} \delta_t^{t+1})||_2 && (12)
\end{aligned}$$

Optimization window For long trajectories, computing the gradients $\nabla_P \ell_{train}$ may be unfeasible due to the GPU memory limitations, for both $\ell_{train} = \ell_{RMS}, \ell_{train} = \ell_{MPRMS}$. In this case, we approximate the gradients of the above criteria by splitting the trajectory to non-disjointed optimization windows:

$$\{\{I_t\}_{t=w\cdot l_{step}}^{w\cdot l_{step}+l_{size}}\}_{w=0}^{\frac{L-l_{size}}{l_{step}}} \tag{13}$$

$$l_{step} < l_{size} \tag{14}$$

where we calculate the gradients $\nabla_P \ell_{train}$ separately for each window, and normalize the gradient up to each length of the trajectory l by the number of occurrences of I_l in different windows, which we denote as $Duplicity(l) = 1 + \left\lfloor \frac{l}{l_{step}} \right\rfloor - \left\lfloor \frac{l-l_{size}}{l_{step}} \right\rfloor$. The criteria ℓ_{RMS}, ℓ_{MPRMS} are then essentially approximated as:

$$\begin{aligned}
&\ell_{RMS}(VO(A(\{I_t\}_{t=0}^{L}, P)), \{\delta_t^{t+1}\}_{t=0}^{L-1}) \\
&\quad \simeq \sum_{w=0}^{\frac{L-l_{size}}{l_{step}}} \sum_{l=w\cdot l_{step}}^{w\cdot l_{step}+l_{size}} \frac{\ell_{VO}(VO(A(\{I_t\}_{t=w\cdot l_{step}}^{l}, P)), \{\delta_t^{t+1}\}_{t=w\cdot l_{step}}^{l-1})}{Duplicity(l)} \qquad (15) \\
&\ell_{MPRMS}(VO(A(\{I_t\}_{t=0}^{L}, P)), \{\delta_t^{t+1}\}_{t=0}^{L-1}) \\
&\quad \simeq \sum_{w=0}^{\frac{L-l_{size}}{l_{step}}} \sum_{l=1}^{l_{size}} \frac{\sum_{i=w\cdot l_{step}}^{w\cdot l_{step}+l_{size}-l} \ell_{VO}(VO(A(\{I_t\}_{t=i}^{i+l}, P)), \{\delta_t^{t+1}\}_{t=i}^{i+l-1})}{(l_{size}-l+1)\cdot Duplicity(l)} \qquad (16)
\end{aligned}$$

Notice that the maximal amount of motions multiplications we are differentiating through would be limited by l_{size}, which would be set according to the memory limitation of the used GPU. In addition, the accuracy of the approximation is improved for smaller l_{step}, at the cost of computational overhead.

3 Assignment

Your goal in this project is to produce universal adversarial perturbations which aim to maximize ℓ_{VO} on unseen trajectories. Below we first detail the task and data specifics as well as the code we provide you with. We afterward details several methodologies which you must address in your work.

3.1 Task specifics

VO model. The VO model used in for this project is TartanVO (1), a recent differentiable VO model that achieved state-of-the-art performance in visual odometry benchmarks. Moreover, in order 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 good degree from the velocity in typical autonomous systems.

Data generation In order to accurately estimate the motions by the VO model, we require a photo-realistic renderer. In addition, the whole scene is altered for each camera motion, requiring re-rendereing for each frame. An online renderer is therefore impractical for optimization, in terms of computational overhead. Moreover, an offline renderer is sufficient for our optimization schemes. We therefore produce the data for optimizing the adversarial patches offline. The renderer framework we make use of is Blender (2), a 3D modeling and rendering package. Blender enables the producing of photo-realistic rendered images from a given 3D scene along with ground-truth motions of the cameras. In addition, we produce high quality, occlusion-aware masks, that are then used to compute the homography H of the patch to the camera viewpoints. For 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 in Eq. (2) as well as the ground truth camera motions δ_t^{t+1}. We produced the trajectories in an urban 3D scene, as it describe a scenario where GPS reception or accuracy is poor and autonomous systems rely more heavily on visual odometry for navigation purposes. The patch is then positioned on a square plane at the side of one of the buildings, in a manner that resembles a large advertising board. An example data frame is depicted in figure 1.

Data specifics We provide you with 50 trajectories with constant 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 contains 8 frames at 30 fps. The trajectories are evenly divided between 5 initial positions. We use a camera with horizontal field-of-view (FOV) of 80° and 640×448 resolution. Your submitted perturbations will be afterwards tested on similarly produced trajectories. The patch is a $30[m]$ square, occupying under the above conditions an average FOV over the trajectories with range of $7.1\% - 7.6\%$, and mean 7.3% area of the images.

Provided code We provide you with code implementing the previously detailed method of producing and testing adversarial perturbations. This implementation will serve as baseline to test the performance of your adversarial perturbations.

3.2 Implementations

You are tasked with improving the produced adversarial perturbations. In your report you must choose and implement approaches for the following methodologies, you must explain your choice and the motivation for each, and present experiments which test the efficiency of your approach.

3.2.1 Optimization scheme

Choose and implement an optimization scheme for the adversarial attacks. The optimization scheme should detail which data is used to optimize the adversarial attacks and how. Detail how you train the perturbations and how do you evaluate it. As the aim is optimizing the perturbations for unseen data,

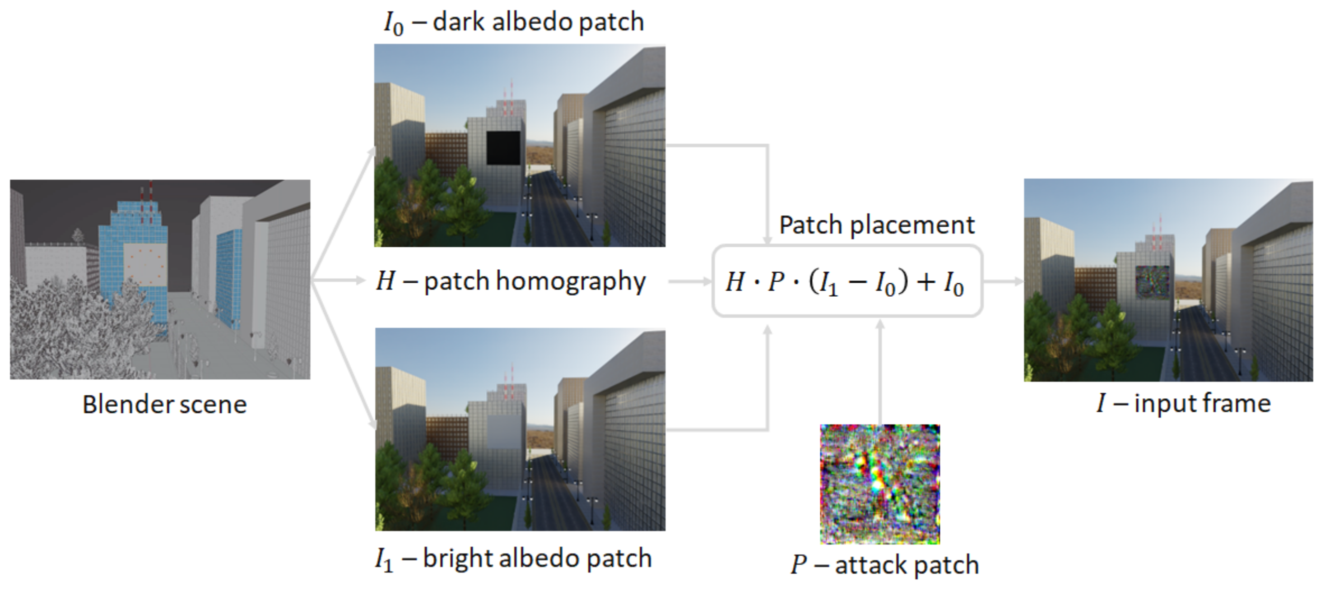
I_0 – dark albedo patch
Blender scene
H – patch homography
Patch placement
$H \cdot P \cdot (I_1 - I_0) + I_0$
I – input frame
I_1 – bright albedo patch
P – attack patch

Fig. 1: Synthetic frame generation. The attack patch P is projected via the homography H and is incorporated into the scene according to the albedo images I_0, I_1.

explain how does your optimization scheme improves the expected generalization of the perturbations to unseen data.

3.2.2 Attack optimizer

Choose and implement an optimizer for your adversarial attacks. The optimizer details the aggregation of gradients over different trajectories and the optimization step size α. consider which optimizer is better suited for the task at hand. Some attacks, such as APGD make use of custom optimizers with adaptive learning rate to better optimize the perturbations, consider making use of such an optimizer.

3.2.3 Optimization and evaluation criteria

Choose and implement an optimization and evaluation criteria for your adversarial attacks $\ell_{train}, \ell_{eval}$. ℓ_{VO} is only directly dependent on the deviation in the physical translation, however the rotation matrices and optical flow produced by the VO model are highly relevant to our task. consider how to make use of such elements. In addition, note that your perturbation aims to maximize ℓ_{VO} on trajectories with the same exact length you are provided with.

4 Submission and grading

4.1 Report structure and evaluation

The following list details what your project report should contain and its impact on the grade:

1. Abstract (6%). Summarize your work. Briefly introduce the problem, the methods and state the key results.
2. Intro (6%). Review the papers relevant to your project. Explain the problem domain, existing approaches and the specific contribution of the relevant paper(s). Also detail the drawbacks which you plan to address.
3. Methods (18%). Explain the original approach as well as your ideas for modifications, additions or improvements to the algorithm/task/domain etc., as relevant. Explain the empirical and/or theoretical motivation for what you are doing.
4. Implementation and experiments (18%). Describe the experiments performed and their configurations, what was compared to what and the evaluation metrics used and why. Explain all implementation details such as model architectures used, data prepossessing/augmentation approaches, loss formulations, training methods and hyper-parameters values.

   Note: You can use pre-existing code in your implementation, but specify what you used and which parts you implemented yourself. The experiments themselves should be unique to your project and implemented by you.
5. Results and discussion (12%). Present all results in an orderly table and include graphs or figures as you see fit. Discuss, analyze and explain your results. Compare to previous works and other approaches for your task.
6. The remaining grade (40%) will be dependent on the performance of your submission in comparison to other groups.

4.2 Evaluation of submitted perturbation

In addition to your report, you will submit a `.png` file containing your best adversarial perturbation. The performance of this perturbation will be tested on the withheld test-set, and compared to perturbation submitted by other groups. The part of your grade dependent on the performance of your submission will than be evaluated according to the average ℓ_{VO} value over the test-set trajectories. You may only submit a single adversarial perturbation for evaluation. Note that the perturbation will be evaluated with the code we have provided you with.

4.3 Submission

Create a `.zip` file titled `proj-id1_id2.zip` (replace `id1`/`id2` with your IDs). The zip file **must** include:

1. A single PDF document, `report.pdf`, containing your project report. It must be structured **exactly** according to the sections listed above.
2. A folder `src/` containing all your code.
3. A `README` text/markdown file, explaining: (i) The structure of the code in the `src/` folder, what is implemented in each package/module; (ii) Steps to reproduce your results using this code, where to get and place the data, how to run all the data processing steps, how to run training and evaluation.
4. A single image named `custom_attack.png` which contains your submitted adversarial perturbation, only this perturbation will be used for the performance evaluation part of your grade.

The zip file **must not** include:

- Training or test data
- Training checkpoints, except when required
- Model parameters, except when required
- Any other unnecessary files

5 General notes and commonly asked questions

5.1 Task

- The required implementations are detailed in Section 3.2.
- In order to receive full grade for the non-competitive part of the assignment, you must implement the detailed assignments and provide a suitable report explaining the task and your approach to the problem, as detailed in Section 4.1. You are not expected to provide additional theoretical analysis of adversarial attacks or visual odometry unless they are directly related to your approach.

5.2 Environment Setup

- In the attached files you will find a file named "install_pytorch_cupy_3.txt". This file provide an example script for creating a suitable environment via conda.
- Note that the provided code can only be run on GPU.

5.3 Code

- You may change the code as you wish, provided that it aligns with the approach you detailed in your report. Note that you are submitting the code and are expected to provide the implementations you used in your experiments. In addition, for the competitive part of your assignment, all the provided perturbation images will be tested on the unchanged provided code, as detailed in Section 3.2.
- The code relating to each of the assignments in Section 3.2 is detailed below, it is recommended that you provide your implementations in the corresponding files, but you may do so differently, as long as you detail in your report where you have chosen to provide said implementations.
- For the high-level optimization scheme implementation (Section 3.2.1) The concerning code is run_attacks.py. You are expected to implement a method similar to the provided "run_attacks_train", which implements the high-level scheme of the optimization process.
- For the implementation of the attack optimizer (Section 3.2.2) The concerning code is attacks\attack.py and attacks\pgd.py, where the first is the base class for adversarial attacks optimizers and the second is an implementation of a basic PGD attack. You are expected to implement a class similar to the provided PGD attack, which implements the attack optimizer.
- For the implementation of the attack criterion (Section 3.2.3) The concerning code is loss.py. You are expected to extend or inherit the "VOCriterion" class with additional criteria concerning the optical flow, the $3D$ rotation and the distance from the adversarial patch. You may also implement additional criteria over the $3D$ translation.

5.4 Recommended papers

- Physical Passive Patch Adversarial Attacks on Visual Odometry Systems (3). This paper discuss the same task as this project and may provides additional insight to your assignment.
- Explaining and Harnessing Adversarial Examples (4).

5.5 common environment errors

- I encountered the error "ModuleNotFoundError: No module named 'cv2'". This is an environment problem, "opencv-python" is not installed correctly. Please follow the provided environment instillation scheme in order to correctly install the environment.
- I encountered the error "ModuleNotFoundError: No module named 'kornia'". This is an environment problem, "kornia" is not installed correctly. Please follow the provided environment instillation scheme in order to correctly install the environment.
- I encountered the error "ModuleNotFoundError: No module named 'cupy'". This is an environment problem, "cupy" is not installed correctly. Please follow the provided environment instillation scheme in order to correctly install the environment.
- I encountered the error "ImportError: cannot import name 'PILLOW_VERSION' from 'PIL'". This is an environment problem, please follow the provided environment instillation scheme in order to correctly install the environment.

5.6 commonly used run parameters

- "–seed n" : set the random seed to be n, may be useful for experiments comparison.
- "–model-name tartanvo_1914.pkl" : This is the used VO model, and should not be changed.

- ”–test-dir ”VO_adv_project_train_dataset_8_frames”” : This is the path to the data folder.
- ”–preprocessed_data : use the prepossessed data from previous runs, this will avoid processing the data for each run.
- ”–batch-size 1” : number of trajectories to be loaded in each batch, should not be changed.
- ”–worker-num 1” : number of workers to load the data, as the data size is substantial, a larger number of workers may cause memory issues.
- ”–max_traj_datasets n” : limits the loaded trajectories datasets to first n folders.
- ”–max_traj_num n” : limits the loaded trajectories from each datasets to first n trajectories.
- ”–max_traj_len n” : limits the loaded trajectories to first n frames.
- ”–save_...” : save flags which controls which results are saved, the available flags are: save-flow, save_pose, save_imgs, save_best_pert, save_csv.
- ”–attack: type of attack to use, default is none, in which only the clean output will be produced. Other types of attacks available in provided code are ”pgd” which optimizes a PGD attack and ”const” which produces an attack based on a given image.
- ”–load_attack: path to load image for ”const” attacks.
- ”–attack_k: number of iterations for attack optimization, it is recommended to use 100 iterations.
- ”–alpha” : attack optimization step size.
- ”–eps 1” : optimization norm limitation, should not be changed.
- ”–attack_t_crit, –attack_rot_crit, –attack_flow_crit, –attack_target_t_crit, –attack_t_factor, –attack_rot_factor, –attack_flow_factor, –attack_target_t_factor” : parameters used for enabling different functionalities of the loss. The output of the VOcriterion loss class is then a weighted sum of the provided criterions, each multiplied by the corresponding factor.

References

[1] Wang, W., Hu, Y., Scherer, S.: Tartanvo: A generalizable learning-based vo. arXiv preprint arXiv:2011.00359 (2020)

[2] Community, B.O.: Blender - a 3D modelling and rendering package. Blender Foundation, Stichting Blender Foundation, Amsterdam. (2018)

[3] Nemcovsky, Y., Yaakoby, M., Bronstein, A.M., Baskin, C.: Physical passive patch adversarial attacks on visual odometry systems. arXiv preprint arXiv:2207.05729 (2022)

[4] Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)