Universal Adversarial Perturbations on Visual Odometry Systems

Tal Ifargan and **Liron Gibli**
Technion - Israel Institute of Technology
{talifargan | gibli}@campus.technion.ac.il

Abstract

In this work we are trying to take on the mission of improving an universal adversarial attack on Visual Odometry (VO) system. We use Projected Gradient Descent (PGD) as a baseline algorithm. We suggest a algorithm that uses momentum, Nestrov adaptive gradient step update and an adaptive method to replace the sign used in PGD and other algorithms. We also add rotation and flow losses that are used in tartanVO and other VO systems to measure the success of the VO in predicting the trajectory as part of our training. We also incorporated data augmentation to test and improve the model universality, which in the end did not work as expected. We were able to improve the out-of-sample results in comparison to the baseline algorithm and thus we are able to tell that our improvements to the algorithm can present better universality, i.e., creating perturbations that might have the ability to attack models in different environments and settings.

1 Introduction

In computer vision, VO is the mission of inferring the motion between two viewpoints. TartanVO focuses monocular VO problem (Wang et al., 2020) and uses loss function that take in account translation, rotation and optical flow.

Our mission is based on (Nemcovsky et al., 2022), In this work Nemcovsky et al. defines the idea of universal adversarial patch attacks and uses PGD algorithm with l_{inf} norm limitation in order to optimize the attack patch and create successful universal attack patches.

There are several other optimization algorithms that were suggested in the area of adversarial attacks, mainly for attacking deep learning models that were developed for classification tasks but we found them very useful in our optimization scheme. One of the these algorithms is the Nesterov Accelerated Gradient (Lin et al., 2019), we elaborate on this method in the following sections and describe the parts of this method that were exploited to improve our results. In their paper the authors also show that besides of improving the attacks on classification models their method can also be used for attacking VO systems. In order to further improve the baseline PGD algorithm we also thought about the sign update step and its limitations and found out that our intuition about using more complicated and adaptive technique was addressed by (Yuan et al., 2021). Another improvement of the optimization scheme came from an idea we have learnt and implemented during the course, momentum, which ofcourse was already implemented before by (Dong et al., 2017) and was able to help the team win a Kaggle challenge in 2017.

Universal perturbations for attacking deep learning models and VO systems in particular is an area still in active research and one that has the potential to improve models stability and susceptibility to malicious attacks.

2 Methods

2.1 Optimization Scheme

The data that was used to optimize the adversarial attacks is the original dataset we were given in addition to augmented data we have created that resembles the method that was suggested by (Gomez-Ojeda et al., 2018). The idea behind these augmentation is to achieve better generalization of the attack by training the attack patch on variety of scenarios but still maintain the original scheme of trajectories with constant length and same data specifics.

We train the perturbations using a train and test set. This will help in understanding the generalization capability (in terms of attack universality) of the attack perturbation we are creating because the perturbation is tested on data which it hasn't trained on.

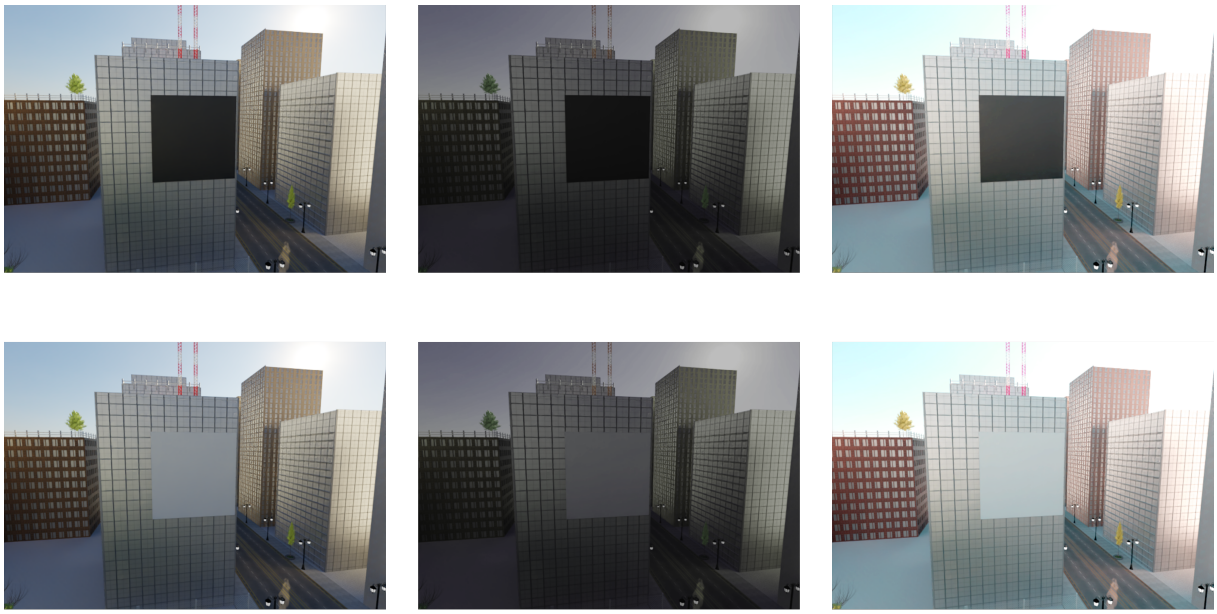

Figure 1: Augmentations. We used augmentations to represent different illumination condition in order to improve the attack patch universality.

2.2 Attack Optimizer

The attack optimizer we chose to adapt in our system was the SINI (Lin et al., 2019) with APAA (Yuan et al., 2021) with a fixed scaling factor. Notice that we are using the Nestrov method (Lin et al., 2019) in order to calculate the gradient, in this method we are looking one step ahead and taking the gradient with respect to the "future" perturbation which is found to have faster convergence in general and also specifically in our case.

2.3 Optimization and Evaluation Criteria

The optimization criteria we implemented comprised of three losses:

- The translation loss $\ell_{translation} = \ell_{RMS}$ is the cumulative translation error loss that was already implemented in the code.
- The rotation loss $\ell_{rotation}$ is the L_2 distance squared (MSE) between the model predicted quaternion q and the ground truth quaternion q_{gt}.
- The flow loss ℓ_{flow} is the MSE loss between the predicted flow F and the ground truth loss F_{gt}

For evaluation we only used the translation loss $\ell_{translation}$ because we found that it is a reliable indicator on the attack success, in addition we saw that in (Wang et al., 2020) they found that the translation loss was responsible for the most significant gap in the model loss.

3 Implementation and Experiments

3.1 Implementation Details

As we mentioned in the last section we used augmentation in order to improve the attack generalization, although eventually the algorithm preformed better without the augmentations we have created. The augmentations included changes in brightness, contrast, gamma, hue and saturation. We had created another 100 additional trajectories, these trajectories has the same meta-data, i.e., the ground truth intrinsic data, trajectory length and camera positions and angles as the original trajectories in the dataset but with different illumination conditions, see Figure 1.

We have implemented train and test split in order to use the train set for training and the test set for evaluation. The data is arbitrarily split into 80% for train and 20% for test. In addition we also implemented k-fold Cross-Validation to tune the model hyperparameters and choose the best performing ones.

The change we incorporated in the attack optimizer is actually manifested in the optimization step of the PGD. The optimization step in iteration k is now:

$p_k^{nes} \leftarrow p_k + \mu \cdot \alpha \cdot g_k$
$g \leftarrow 0$
for $i = 1$ to N_{train} **do**
    $\hat{y}_{train,i} \leftarrow VO(A(x_{train,i}, p_k^{nes}))$
    $g \leftarrow g + \nabla p_k^{nes} \ell_{train}(\hat{y}_{train,i}, y_{train,i})$
end for
$g_{k+1} \leftarrow \mu \cdot g_k + \frac{g}{\|g\|_1}$
$p_{k+1} \leftarrow clip(p_k + \alpha \cdot \gamma \cdot g_{k+1})$

Where μ is the momentum parameter which comes from the idea that using the accumulated ascent direction can help avoiding local maximum (Dong et al., 2017), α is the learning rate as given in the PGD attack and γ is the scaling factor (Yuan et al., 2021) that replaces the sign update rule in the PGD attack.

As we mentioned in the last section the loss we have implemented in the model was comprised of three different losses:

- The translation loss $\ell_{translation}$:

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

- The rotation loss $\ell_{rotation}$:

$$\ell_{rotation} = \|q - q_{gt}\|^2 = \|q\|^2 - 2(q \cdot q_{gt}) + \|q_{gt}\|^2 = 1 - 2(q \cdot q_{gt}) + 1 = 2(1 - q \cdot q_{gt}) \quad (2)$$

- The flow loss ℓ_{flow}:

$$\ell_{flow} = \|F - F_{gt}\|^2 \quad (3)$$

3.2 Experiments and Evaluation

In order to find the best hyperparameters we used cross-validation, The hyperparameters we have tuned included:

- Number of epochs - we have tested several different values in the range of 10 - 100 and we have found that we can test our hyperparameters over 10 epochs and get a satisfactory result.

- Alpha - we have tested different alpha values that were in different order of magnitude, 0.005, 0.05 and 0.5. The one that worked best was 0.05.

- Mu - as we have explained in the last section mu is the momentum parameter, as presented by (Dong et al., 2017), the choice of picking a value of 1 was that this value expresses the idea we take a step in the cumulative direction of all the previous steps.

- Gamma - we used the same value $gamma = 8$ that worked best for (Yuan et al., 2021) and actually implementing this method in our model was able to improve the attack results significantly.

After finding the best hyperparameters we have trained an attack perturbation with the chosen parameters on all the train data and tested the result of the attack on the train and test data. the evaluation of the attack was done using the RMS criterion that was presented in the Project Guideline. We have also compared our attack ℓ_{RMS} to the baseline PGD attack ℓ_{RMS} in order to see if we improve over the baseline.

4 Results and Discussion

As mentioned in the last section we preformed hyperparameters search using kfold cross validation in order to find the best preforming params. The results are presented in Figure 2.

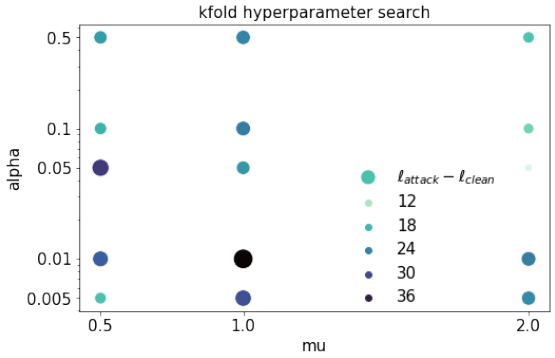

Figure 2: Kfold cross validation. The scatter shows the differences between our attack loss ℓ_{attack} and clean loss ℓ_{clean}.

The best hyperparameters were $mu = 1$ and $alpha = 0.01$ which used us for training the attack.

We have trained our perturbation using the original data and the augmentations we have created and tested the resulted perturbation performance. The results are presented in terms of loss for each trajectory where the loss includes all the terms we have implemented with equal weighing. Figure

3 and Figure 4 below shows the results with the augmentations.

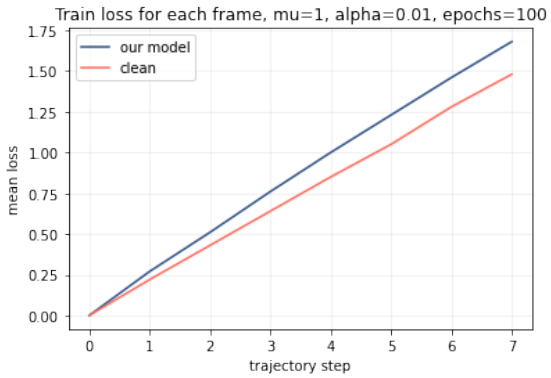

Figure 3: Train loss for each frame with augmentations.

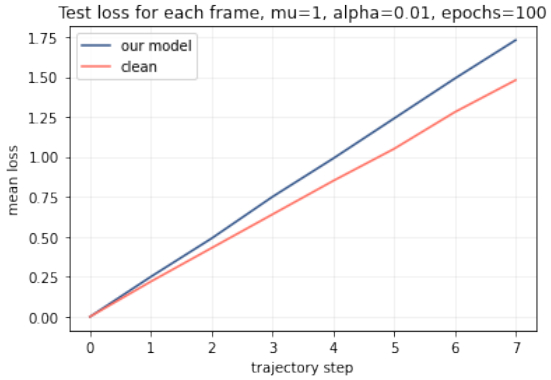

Figure 4: Test loss for each frame with augmentations.

The results we got were not satisfactory and we decided to test if training the perturbation without the augmentations will improve them, the results are presented the same way as before with addition to the comparison to the baseline algorithm.

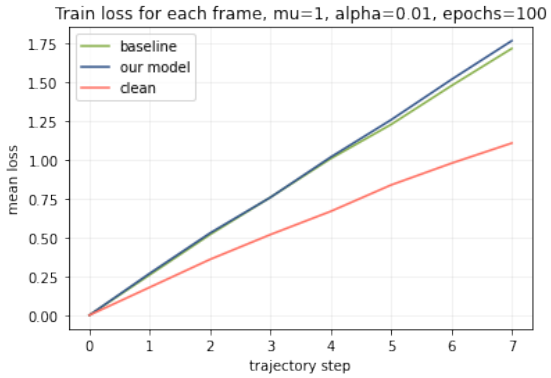

Figure 5: Train loss for each frame.

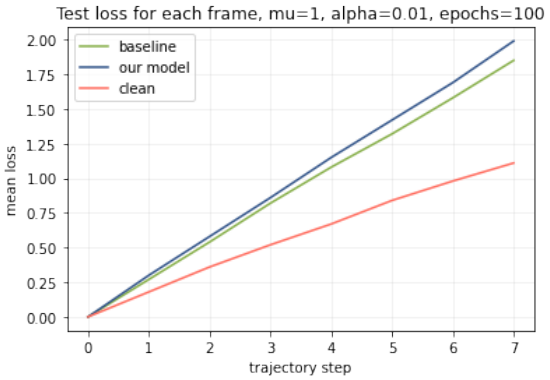

Figure 6: Test loss for each frame.

We can see that our algorithm was able to produce result that was especially good at out-of-distribution which is a very favorable result.

References

Y Dong, F Liao, T Pang, H Su, J Zhu, X Hu, and J Li. 2017. Boosting adversarial attacks with momentum. arxiv. *arXiv preprint arXiv:1710.06081*.

Ruben Gomez-Ojeda, Zichao Zhang, Javier Gonzalez-Jimenez, and Davide Scaramuzza. 2018. Learning-based image enhancement for visual odometry in challenging hdr environments. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 805–811. IEEE.

Jiadong Lin, Chuanbiao Song, Kun He, Liwei Wang, and John E Hopcroft. 2019. Nesterov accelerated gradient and scale invariance for adversarial attacks. *arXiv preprint arXiv:1908.06281*.

Yaniv Nemcovsky, Matan Yaakoby, Alex M Bronstein, and Chaim Baskin. 2022. Physical passive patch adversarial attacks on visual odometry systems. *arXiv preprint arXiv:2207.05729*.

Wenshan Wang, Yaoyu Hu, and Sebastian Scherer. 2020. Tartanvo: A generalizable learning-based vo. *arXiv preprint arXiv:2011.00359*.

Zheng Yuan, Jie Zhang, and Shiguang Shan. 2021. Adaptive perturbation for adversarial attack. *arXiv preprint arXiv:2111.13841*.