Institute of Visualization and Interactive Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Master's Thesis

Attacking a Defended Optical Flow Network

Alexander Lis

Course of Study:	Informatik
Examiner:	Prof. Dr. Andrés Bruhn
Supervisor:	Jenny Schmalfuß, M. Sc.
Commenced:	2021-10-20
Completed:	2022-04-20

Abstract

Deep Neural Networks for optical flow estimation achieve exceeding performances on published datasets. However successful and practically applicable patch attacks in the past and the lack of robustness guarantees for published networks demonstrate the importance to further study their resilience against manipulated inputs. The recent desire to deploy optical flow networks in security-critical applications like autonomous driving or robot navigation further amplifies this problem. Recently, the suitability of localized pre-processing defenses against adversarial patch attacks for optical flow networks was examined and a specialized defense was proposed. However an extensive robustness assessment of the defended systems using adaptive attacks was missing. Furthermore results about adaptive attacks on defended networks are currently rather restricted to classification networks.

In this thesis we devise adaptive adversarial patch attacks against the optical flow network FlowNetC when it is defended by the specialized defenses Inpainting with Laplacian Prior (ILP) and Local Gradients Smoothing (LGS). We provide empirical evidence that our adaptive white-box attacks increase the efficiency of injected patches significantly compared to the attacks considered in their initial evaluation. Our attacks introduce serious distortions in the flow field estimation of defended networks. Additional contributions are the implementation of a flexible training pipeline and the reimplementation of the Inpainting with Laplacian Prior defense according to its description in the original publication.

Kurzfassung

Neurale Netwerke zur Bestimmung des Optischen Flusses erreichen Bestleistungen bei Experimenten mit publizierten Datensätzen. Allerdings betonten erfolgreiche und praktische Manipulationsangriffe auf die Eingabebilder in der Vergangenheit, sowie das Fehlen von Robustheitsabschätzungen für publizierte Netzwerke, die Relevanz von weiteren Untersuchungen zur Verlässlichkeit von diesen Netzwerken. Diese sind außerdem eine wichtige Voraussetzung für die Nutzung dieser Algorithmen in sicherheitskritischen Anwendungen. Die Eignung von Local Pre-processing Verteidigungen zum Schutz dieser Netzwerke gegen Adversarial Patch Angriffe wurde kürzlich untersucht und eine speziell auf Netzwerke zur Bestimmung des Optischen Flusses angepasste Verteidigung entwickelt. Allerdings wurde die Robustheit der resultierenden Systeme nicht gegen adaptive Angriffe evaluiert, die Informationen über die Verteidigungen ausnutzen. Außerdem beschränken sich bisherige Ergebnisse zu adaptiven Angriffen vorwiegend auf Netzwerke zur Bildklassifikation.

Diese Arbeit entwickelt adaptive Adversarial Patch Angriffe gegen die Pre-processing Verteidigungen Inpainting with Laplacian Prior (ILP) und Local Gradients Smoothing (LGS) im Kontext der Bestimmung des Optischen Flusses mit dem Netzwerk FlowNetC. Unsere Experimente deuten darauf hin, dass unsere adaptiven white-box Angriffe die bisherigen Robustheitsabschätzungen der Verteidigungen signifikant korrigieren. Weitere Beiträge dieser Arbeit sind die Implementierung einer flexiblen Pipeline zum Trainieren von Adversarial Patches sowie die Reimplementierung der Inpainting with Laplacian Prior Verteidigung nach der Beschreibung in der Veröffentlichung.

Contents

1 Introduction

Deep Neural Networks achieve outstanding performances in diverse visual tasks including image and video classification [34, 37], object detection [64] and face recognition [77]. Recently, they also outperformed traditional approaches in the domain of optical flow estimation [78]. The increasing relevance of neural networks in optical flow estimation is reflected in their growing number of top performing algorithms on the public scoreboards for the established datasets Sintel [13] and KITTI15 [48]. As a result, optical flow networks are considered for deployment in security-critical applications like autonomous driving [63], street sign classification [19] or robot navigation [89]. However prior work indicated that it is difficult to approximate the robustness of neural networks [18].

Attacks are one way to approximate upper bounds on a network's performance measure [18]. Input manipulation attacks intend to confuse a network by manipulating its input image prior to processing [63]. These attacks were originally conceived for the image classification problem, however recently they are also being considered in the context of optical flow estimation [3, 63]. Adversarial patch attacks are a particularly practical instance of input manipulation attacks [10, 63]. They introduce strong and static perturbations in a confined region around the patch's location and can thus be applied physically [63]. Ranjan *et al.* [63] demonstrated that some optical flow networks were vulnerable to adversarial patches.

Pre-processing defenses were developed to protect neural networks against input transformation attacks. As a response to the attacks on optical flow networks by Ranjan *et al.* [63], Anand *et al.* [3] examined the efficiency of two pre-processing defense mechanisms against adversarial patch attacks in the context of optical flow networks. The first considered defense is called Local Gradients Smoothing and was originally conceived by Naseer *et al.* [52] to protect classification networks. The second algorithm evaluated is called Inpainting with Laplacian Prior and is a pre-processing defense by Anand *et al.* [3] that is specifically developed for optical flow networks. The adversarial model employed for the robustness assessment of the defended systems was very similar to the original attack by Ranjan *et al.* [63]. Anand *et al.* [3] determined that the defenses increase the network's robustness significantly against their attacker when an optical flow network is a component in an action recognition pipeline [3].

However previous research has already indicated that there seems to be no single adversary that yields an effective attack without including additional information about the target network [80]. Additionally, defenses can obfuscate gradient information and as a result only appear to be robust against defenses even though there might exist very effective attacks that require a specific optimization approach [4]. For classification networks it is therefore already a standard to test defenses against attacks that specifically employ knowledge about them [80]. Consequently, the reuse of existing attacks as performed in the publication by Anand *et al.* [3] is a strong indication that the applied attacks do not use the full adversarial potential available [81]. Subsequently, the evaluation might not reflect the true worst case performance under an adversarial patch attack. Tramèr *et al.* [80] published guidelines that identify recurrent principles which frequently lead to successful adversarial attacks. Similarly Athalye *et al.* [4] provide guidelines to improve the convergence of white-box adversaries that employ gradient based methods on networks that conceal information about the gradient.

In this work we propose adaptive attacks against the optical flow network FlowNetC when it is defended by the local pre-processing defenses ILP or LGS. For the formulation of our adversaries, we employed the methodology for the development of adaptive attacks published by Tramèr *et al.* [80]. Furthermore we considered approaches to recover obfuscated gradients during patch optimization by Athalye *et al.* [4] and considered general guidelines about adaptive attacks on classification networks [14]. Moreover our attacks take special consideration of previous work on the robustness of optical flow networks [3,63,67,68]. Additionally our attacks are built on and extend previously published attacks in the context of image classification by Chiang *et al.* [18] on LGS [52] and Digital Watermarking [27].

1.1 Thesis Structure

The remainder of this work is structured as follows. The second chapter summarizes related work. It focuses on delineating research that is connected to robustness estimation of optical flow networks and introduces attacks, defenses and evaluation procedures. The third chapter presents models that describe our attacks and the defenses. Furthermore detailed specifications of the attacked pre-processing defenses are provided and adaptive adversaries are derived. The fourth chapter then reports our experimental arrangement and presents our evaluation results. Lastly the sixth chapter provides an outlook about potentially relevant future work and finally summarizes essential insights.

2 Related Work

The following sections discuss publications which are related the development of adaptive image input manipulation attacks on optical flow networks. Generally, robustness estimation of neural networks via input manipulations is a field that was traditionally studied for image classification networks [76]. Consequently, our summary regularly references concepts that were initially designed for classification networks specifically. However, many ideas generalize well to different network classes including optical flow networks [3, 63, 67, 68].

The first subsection will introduce the optical flow problem and mention milestone neural network architectures. Secondly, we introduce traditional approaches to estimate the performance of optical flow networks. Thirdly, we summarize contributions to perturbation attacks, a term that we use to describe a generalization of attacks like adversarial examples and patches. The fourth section summarizes approaches to examine the robustness of optical flow networks via input transformation attacks. In the fifth section we examine concepts to defend neural networks. Lastly, we summarize published guidelines about the design of attacks against defended networks and about the development of robust defenses.

2.1 Optical Flow

Horn and Schunck [28] canonically defined the optical flow of two sequential images to be the movement of brightness patterns between them. Similarly to the interpretation by Ranjan *et al.* [63] we consider the optical flow F between two sequential images I_1 and I_2 to be the displacement field F that fullfills some photometric assumption. Considering as a demonstration the brightness constancy assumption [28] then requires for a location (x, y) and the respective displacement $F(x, y) = (u, v)$ at this location

$$I_1(x, y) = I_2(x + u, y + v). \tag{2.1}$$

Traditionally this problem has been formulated as an energy minimization problem [28]. Teed *et al.* [78] concisely summarized existing approaches to solve the optical flow problem and milestone network architectures that we want to mention here. Besides, the problem has also been approached as a discrete optimization problem [17, 49]. Dosovitskiy *et al.* [20] demonstrated that this correspondence problem could also be approached as an optimization problem for neural networks. Specifically, they proposed two types of convolutional neural network architectures that they termed FlowNetS and FlowNetC [20]. Especially the approach of FlowNetC to introduce an explicit correlation layer was an important contribution that influenced many later architectures [78]. Notable subsequently published optical flow networks aimed to improve the performance by combining FlowNetS and FlowNetC units and a subsequent fusion operation with a related network [30] or to choose a network architecture that is aware of spatial pyramids [62]. Additional improvements could be achieved by introducing warping and cost volumes into the network's architecture [75]. Recently, inspired by the traditional energy minimization approach, Teed *et al.* [78] published the Recurrent All-Pairs Field Transforms network (RAFT). It is characterized by the combination of a feature encoder with a 4D correlation volume and a recurrent iterative refinement of the flow estimate [78].

2.2 Performance Evaluation of Optical Flow Networks

Baker *et al.* [6] published a comprehensive summary about evaluation methodology for optical flow estimators. They determined two approaches that are widely used in literature to estimate the error between a ground truth flow field $F_{gt} \in \mathbb{R}^{2\times H\times W}$ with height H and width W and an algorithmic prediction $F \in \mathbb{R}^{2\times H\times W}$. Intuitively, these measures provide an estimate for the distance between optical flow fields within metric spaces [67].

The Endpoint Error (EE) [6] represents the euclidean distance between the endpoints of two individual vectors. Typically one compares the ground truth flow vector (u_{gt}, v_{gt}) and a predicted flow vector (u, v):

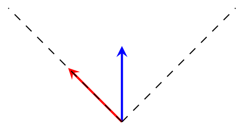
$$EE\left(\begin{pmatrix} u_{gt} \\ v_{gt} \end{pmatrix}, \begin{pmatrix} u \\ v \end{pmatrix}\right) = \sqrt{(u-u_{gt})^2 + (v-v_{gt})^2}. \tag{2.2}$$

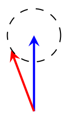
To estimate the performance of an entire flow field prediction F with respect to a ground truth F_{gt} this measure is generalized to the average endpoint error (AEE) [6]. Let from now on $\mathbb{N}_n = \{k \in \mathbb{N} \mid k < n\}$. Then this yields

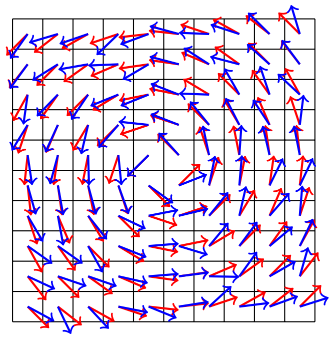
$$AEE(F_{gt}, F) = \frac{1}{W \cdot H} \sum_{(y,x)\in\mathbb{N}_H\times\mathbb{N}_W} EE(F_{gt}(y,x), F(y,x)). \tag{2.3}$$

Secondly, the Angular Error (AE) estimates the smaller angle between the ground truth and the prediction vectors, however generalized in a three dimensional space for an increased stability [6].

$$AE\left(\begin{pmatrix} u_{gt} \\ v_{gt} \end{pmatrix}, \begin{pmatrix} u \\ v \end{pmatrix}\right) = \cos^{-1}\left(\frac{1 + uu_{gt} + vv_{gt}}{\sqrt{1+u^2+v^2}\sqrt{1+u_{gt}^2+v_{gt}^2}}\right).$$


(a) Intuitive visualization of the CS distance measure. Assuming the blue arrow represents the ground truth flow prediction the dashed gray lines indicate predictions that result in the same CS score compared to the red arrow prediction.


(b) Intuitive visualization of the EE distance measure. Assuming the blue arrow represents the ground truth flow prediction the dashed gray lines indicate predictions that result in the same EE score compared to the red arrow prediction.


(c) Visualization of the generalization from single prediction to numerous predictions that is necessary to compare flow fields.

Again, this notion is generalized to the Average Angular Error (AAE) [6] to compare flow fields

$$AAE(F_{gt}, F) = \frac{1}{W \cdot H} \sum_{(y,x) \in \mathbb{N}_H \times \mathbb{N}_W} AE(F_{gt}(y,x), F(y,x)). \tag{2.4}$$

Ranjan *et al.* [63] additionally proposed the Average Cosine Similarity (ACS) performance measure in the context of adversarial patch attacks on optical flow networks. They employed it as a score to optimize for successful flow manipulations $\hat{F}$ compared to an unattacked flow F. Similarly to the AAE distance it formalizes the intuition about how aligned the individual prediction directions are. The actual PyTorch implementation that the authors included a small offset to increase stability. It is defined as

$$ACS(F, \hat{F}) = \frac{1}{W \cdot H} \sum_{(y,x) \in \mathbb{N}_H \times \mathbb{N}_W} \frac{F(y,x) \cdot F(y,x)}{\max\left(\|F(y,x)\| \, \|\hat{F}(y,x)\|, 10^{-8}\right)} \tag{2.5}$$

Schmalfuß *et al.* [67] mention the Mean Squared Error (MSE) as a potential alternative to the AEE which is more robust with respect to differentiability. However as the authors mentioned [67], the measure introduces a stronger sensitivity to outliers in exchange.

$$MSE(F, \hat{F}) = \frac{1}{H \cdot W} \sum_{(y,x) \in \mathbb{N}_H \times \mathbb{N}_W} \|F(y,x) - \hat{F}(y,x)\|_2^2 \tag{2.6}$$

Public benchmarks use among other accuracy focused measures like F1,F3,F5 error the AEE as a performance measure to evaluate and compare optical flow algorithms [13, 48]. Schmalfuß *et al.* [67] determined that current optical flow evaluation focuses rather on accuracy and largely neglects robustness.

Notable public datasets in Optical Flow estimation are KITTI15 [48] and Sintel [13]. KITTI15 intends to comprise sequential image pairs which are representative for the driving domain [48] while Sintel focuses on synthetic images which contain difficult phenomenons like motion blurring, large motions and occlusions [13]. Consequently computing the AEE over one of those datasets D for an optical flow network N aims to approximate the true expectation value of the networks performance in the respective domain $\mathcal{D}$ [13, 48]. Assuming a dataset D comprises image pairs I_1, I_2 and the associated ground truth flow fields F_{gt}. Let furthermore N denote an optical flow algorithm that outputs a flow estimate derived from two input images. Then this can be described as

$$\frac{1}{|D|} \sum_{((I_1,I_2),F_{gt})\in D} AEE(F_{gt}, N(I_1, I_2)) \approx \mathbb{E}_{((I_1,I_2),F_{gt})\sim\mathcal{D}}\left[AEE(F_{gt}, N(I_1, I_2))\right] \tag{2.7}$$

We consider Optical Flow Networks that take the approach to learn the prediction of the brightness movements between sequential images from annotated datasets in a supervised learning fashion [20]. Public datasets are partitioned into a training set that is provided with ground truth information and a separate test set to prevent an overfitting to the training data [13, 48].

2.3 Image Input Manipulation Attacks

Publications examining input manipulation attacks on image processing networks like adversarial patches or examples regularly approach the task to find the strongest respective image manipulation as a constrained optimization problem [67, 76]. Similarly as in existing works [40, 67] we consider an unified notion of adversarial patches and adversarial examples that we call adversarial perturbations.

Originally these attacks have been approached for image classification networks [5, 10, 33, 76]. Szegedi *et al.* [76] were the first who noticed that classification networks could be confused by small input transformations that did not affect human classification decisions, if they were noticed by humans at all. This observation raised fundamental questions about the robustness of classification networks [76]. Later this was observed by researchers in various different domains including optical flow [63, 67, 68], stereo disparity estimation [84] or video classification [32]. In the following years research has built upon this idea in various respects that we want to delineate in the following.

At first we consider the perturbation representations proposed in literature. Next we summarize important concepts to constraint the introduced perturbations and their implications. Thirdly, we discuss differentiations with respect to the goal of a perturbation. We also consider research with respect to a perturbations transferability, threat model, and optimization procedures. We aim to present our personal view on adversarial perturbations with a focus on concepts that are transferrable to optical flow networks.

2.3.1 Perturbation Representation

There are differences between attacks with respect to how the parametric representation of a perturbation P is subsequently used to introduce changes to an image. Publications that consider small brightness changes rather use additive changes [24, 76].

$$\hat{I} = I + P$$

Publications examining the effects of practically unlimited brightness changes in spatially confined regions, restricted by a mask M, often model their noise such that it replaces the original image content in the affected regions [10, 33, 63]

$$\hat{I} = (1 - M) \odot I + M \odot P$$

Wortman [85] published an attack that additionally considers a patches transparency with the intention to hide such perturbations from human observers. Assuming alpha blending using a mask α, the patch perturbation P injection would then be modeled as

$$\hat{I} = (1 - \alpha) \odot I + \alpha \odot P$$

Athalye et al [5] also consider to represent the perturbation in a different color model for additional constraints about their visibility. Specifically they considered LAB as it provides a perceptibly uniform color space [46]. Assuming the perturbation in another color space is P' and the suitable transformation to the target color space is f then we would have to require for the equations above

$$P = f_{lab \to rgb}(P')$$

Conventionally the resulting images I are assumed to be scaled to a finite interval e.g. $[0, 1]$ or $[0, 255]$ for neural networks. Thus to directly generate restricted image perturbations, Carlini and Wagner [16] explicitly propose a suitable variable substitution $P = f(P')$ with a suitable function f which has a global domain like $\mathbb{R}^{h \times w}$ and a limited codomain e.g. $[0, 1]$. In this way they incorporated the implicit constraint on the codomain of the patch into their perturbation representation model [16]. As an specific example the authors propose a function that is based on the tanh function

$$P = \frac{1}{2} \left(\tanh(P') + 1 \right) \tag{2.8}$$

2.3.2 Perturbation Constraints

There are constraints with respect to the shape and brightness of the perturbation distinguished [10, 24]. For image sequences Ranjan *et al.* [63] also consider constraints about perturbation changes over time between the images. The following paragraphs discuss these constraints and related literature in more detail.

Limitations regarding possible brightness differences are often specified via an upper bound ϵ on the p-norm induced distance [16, 70] between the original image I and the perturbed image $\hat{I}$

$$\|I - \hat{I}\|_p < \epsilon, \quad \text{with } \|I\|_p = \left(\sum_{i,j} |I(i,j)|^p \right)^{\frac{1}{p}} \tag{2.9}$$

Very often the (optical flow) network requires its input to be located in a (possibly discrete) range $[a, b]$ for instance $[0, 255]$. This results in a codomain constraint [7, 16]

$$\hat{I} \in [0, 255]^{H \times W} \quad \text{or even } \hat{I} \in \{0, 1, \dots, 254, 255\}^{H \times W}$$

Common choices for p in perturbation constraints include 0,1,2 and ∞. The L_0 distance is often used to introduce quantitative spatial constraints on the patch as it specifies the possible number of pixel manipulations [58, 70]

$$|I|_0 = |\{(i,j) \mid I(i,j) \neq 0\}|$$

Papernot *et al.* [57] published their Jacobian based Saliency Map Attack (JSMA) that derives perturbations constrained under the L_0 distance. Additionally Papernot *et al.* [58] use this metric in the robustness argumentation of a preprocessing defense which is called Distillation.

Biggio et al. [7] considered an L_1 metric to generate adversarial examples for handwritten digit classification.

$$\|I\|_1 = \sum_{(i,j)} |I(i,j)| \tag{2.10}$$

The L_2 norm was already present in early works on adversarial examples in image classification works [29, 76]

$$\|I\|_2 = \sum_{(i,j)} \sqrt{|I(i,j)|^2}. \tag{2.11}$$

The L_∞ constraint is present in literature on global attacks [4, 56, 67, 68]. It specifies the maximally allowed perturbation between the original image I and the perturbed image $\hat{I}$ for any pixel

$$\|I\|_\infty = \max_{(i,j)} |I(i,j)| \tag{2.12}$$

Spatial constraints specify additional assumptions about locations at which the inclusion of a perturbation is allowed [10,33,63]. In one possible interpretation this constraint adds to the perturbation the notion of an origin location $l = (y, x)$ and a patch radius r in an L_2 norm to generate round adversarial patches [10,63]. One possible formal formulation could require image regions outside a p-sphere of radius r to remain static while regions within this square are allowed to be perturbed. For an attacked image $\hat{I}$ and the unattacked version I this can be stated as

$$\forall i, j : \left\| \begin{pmatrix} i \\ j \end{pmatrix} - l \right\|_p < r \Rightarrow I(i, j) = \hat{I}(i, j) \tag{2.13}$$

There are generalized approaches to include colored objects with custom shapes into the image [5,73]. Sharif *et al.* [73] include colored eyeglasses into images to confuse face recognition systems and Athalye *et al.* [5] use 3D printed objects to manipulate image classification networks.

Ranjan *et al.* [63] consider temporal constraints for image sequence inputs I_1, I_2 for optical flow estimation. Specifically they assumed the perturbations to be static over time [63]. In their case this assumption had at least two consequences. Firstly it resulted in a model that was practically applicable [63]. Secondly it simplified the derivation of the adapted flow ground truth $\hat{F}_{gt}$ [63,67]. Schmalfuß *et al.* [67] explicitly distinguish between three additional types of perturbation constraints for optical flow networks. Firstly they consider joint perturbations that are required to apply the same perturbation to both images of an image pair. Secondly disjoint adversarial perturbations train two global perturbations P_1, P_2 each applied to an individual frame I_1, I_2 separately [67]. Thirdly, they distinguish universal adversarial perturbations as an constraint that is orthogonal to the previously mentioned ones [67].

2.3.3 Perturbation Goal

There are already classifications of different adversarial goals published by Carlini *et al.* [14]. A distinction that is clearly reflected in published literature is that adversarial perturbations are used to examine the robustness of neural networks for a wide range of problems. Originally Szegedi *et al.* [76] considered classification networks. Recently their effect is being examined in diversified problem areas including optical flow estimation [63,67,68], stereo disparity estimation [84], monocular depth estimation [90] and video classification [3,32].

Another distinction criterion provides the optimization goal that is specified via a loss function. In optical flow estimation Ranjan *et al.* [63] proposed to use the ACS loss as specified in Section 2.2. This optimization goal was also considered in subsequent works attacking optical flow networks [67,68,85]. Schmalfuß *et al.* [67] additionally proposed the AEE and MSE as potential loss functions in the domain of optical flow estimation.

Similarly to attacks on classification networks, one can distinguish between targeted and untargeted attacks against optical flow networks [67]. Ranjan *et al.* [63] approximated using their adversarial patch attack on neural networks the inverse flow of the unattacked prediction. Schrodi *et al.* [68] trained their codomain constrained global perturbation attacks to result in arbitrarily selected flow fields.

Ranjan *et al.* [63] also introduced the idea to optimize the difference between the unattacked flow field prediction instead of the ground truth flow. According to them a practical benefit of this approach is that it does not require to elaborately gather ground truth data [63]. Additionally, Schmalfuß *et al.* [67] provided further intuitive reasoning to consider robustness and prediction accuracy as different objectives which should be separated.

2.3.4 Perturbation Robustness

Robustness, universality or transferrability in the context of adversarial perturbations describe how invariant a particular perturbation instance is with respect to changes of training parameters [5, 63, 68, 84]. Possible considered parameters are the perturbation location, rotation and scale for adversarial patches [10, 63]. Additionally illumination and perspective are considered explicitly [5]. Also the image content [67, 84] or the target neural network [63] are considered.

Athalye *et al.* [5] published a method to increase an the robustness of a perturbation which is called Expectation over Transformation (EoT). The method aims to increase a perturbations robustness by randomizing specific training parameters during training and introduces updates that approximate the expected loss [5]. They used their method to construct 3D printable objects that were able to confuse an image classification network despite of different illumination conditions, noise and affine transformations. Ranjan *et al.* [63] optimized adversarial patches against multiple optical flow networks jointly and compared their performance with individually trained ones. Schrodi *et al.* [68] developed global codomain constrained attacks on modern optical flow networks that were able to confuse these networks to output almost arbitrary flow fields. Their attacks were however specialized to the image pair instance and thus not universally applicable [68]. Similar results were published by Wong *et al.* [84] for stereo depth estimation and by Schmalfuß *et al.* [67] for optical flow networks. Brown *et. al.* [10] introduced an adversarial patch attack against classification networks that was universal with respect to location and a range of rotational and scaling variations.

2.3.5 Practical Applicability

A notion that is closely connected to universality is the practical applicability in the physical world. Attacks need to provide a sufficient degree of universality in order to be applied in the physical world [5]. Kurakin *et al.* [39] demonstrated that image classification networks could be confused by providing them with adversarial examples displayed on a phone display. Athalye *et al* [5] demonstrated that 3D printed physical objects could be crafted that fooled image classification networks because of their robustness to lighting conditions and perspectives. Evtimov *et al.* [22] published an attack algorithm that computes physically realizable perturbations that confuse road sign classification networks. Sharif *et al.* [73] derived eyeglasses frames that introduced perturbations that confused face recognition systems. Brown *et al.* [10] produced adversarial patches that confused image classification networks and were practically applicable because of their robustness to limited scaling and rotation operations. The patches produced by Ranjan *et al.* [63] were based on the work by Brown *et al.* [10] and are also physically realizable against few optical flow networks. Wortman [85] extended the work by Ranjan *et al.* [63] and considered constraints on the patches alpha value to hide the local perturbations from human observers.

2.3.6 Adversary Knowledge

Carlini *et al.* [14] provide an extensive discussion on adversarial threat models which encompasses the notions of perturbation goals and adversary knowledge specified for this work. Adversarial knowledge describes the information an attacker model is provided with [14]. Carlini *et al.* [4, 14] state that a widespread yet broad distinction is between white-box and black-box attackers. They further argue based on Kerkhoffs' principle [35] that implementations that are considered for deployment should assume that the specification of the algorithm is available to the attacker [14]. Attackers that have access to the target network including its weights and architecture are called white box adversaries [4, 14]. A more special notion that is considered are adversaries that have access to the image beforehand [68, 84]. Adversaries without such internal information are called black-box attacks [4,14]. With respect to black-box adversaries there is also the notion considered in which an adversary may have access to neural networks that are different to the target network but aim to solve the same problem [63]. Athalye *et al.* [4] summarize components that can additionally be specified by the threat model. Their list includes architecture, model weights, training algorithm and training data, test time randomness, query access [4].

2.3.7 Optimization

The idea of the optimization process is to find the perturbation parameters that yield the most successful attack with respect to the adversarial goal [63]. For algorithmic computation this goal is implemented as a differentiable training loss function that is minimized [63]. Assuming white-box information about the network's weights and architecture is provided to the attacker, she is able to use optimization procedures based on backpropagation [66] to improve her perturbation successively [14]. The employed optimization technique is another distinguishing aspect of attacks [24, 63, 76, 84]. Szegedi *et al.* [76] use a constrained L-BFGS algorithm [42] to compute efficient adversarial examples in their initial work.

Goodfellow *et al.* [24] proposed the Fast Gradient Sign Method to derive adversarial examples for additive perturbations of image classification networks in a single update step. For a scalar valued loss function $\mathcal{L}$, an adversarial perturbation P that is applied to an input image I and a target label y they compute their codomain constrained perturbation using

$$\hat{I} = I + P = I - \epsilon\, \mathrm{sgn}(\nabla_I \mathcal{L}(N(I), y))$$

Kurakin *et al.* [39] published the I-FGSM optimization procedure that applies the FGSM procedure iteratively

$$I_0 = I, \quad I_{n+1} = I_n - \alpha\, \mathrm{sgn}\left(\nabla_I J(N(I_n), y)\right) \tag{2.14}$$

The optimization algorithm is used by Schrodi *et al.* [68] for their global attack on optical flow networks. Additionally Wong *et al.* [84] employ it for their global perturbation attack against stereo disparity networks.

Carlini and Wagner [14] and Madry *et al.* [44] use projected gradient descent to optimize their patches. Assuming for simplification that the projection is implemented as a clipping operation to the interval $[0, 1]$ the update states

$$I_0 = I, \quad I_{n+1} = Clip_{[0,1]}\left\{I_n - \alpha\, \nabla_I \mathcal{L}(N(I_n), y)\right\}. \tag{2.15}$$

As already mentioned in Section 2.3.1 Carlini and Wagner [16] proposed the use of reparametrization of the perturbation in order to convert the constraint optimization problem into a global one to be able to use other optimization algorithms such as SGD [65] or ADAM [36].

2.4 Robustness Estimation of Optical Flow Networks

Schmalfuß *et al.* [67] distinguish the prediction robustness from the prediction accuracy and provide a ranking of optical flow networks that considers both measures. Specifically Schmalfuß *et al.* [67] differntiate the robustness an optical flow network with respect to adversarial perturbations from the prediction accuracy that we introduced in Section 2.2. This section summarizes research focusing on the estimation of the robustness of optical flow networks to adversarial perturbations.

Ranjan *et al.* [63] proposed the Zero-Flow test as a means to examine the behavior of an optical flow network in the presence of local patch perturbations. The test can be subdivided into three steps. At first, an image is sampled from random noise and amended with a replicate to form an image pair (I_1, I_2) with $I_1 = I_2$. Next, a second image pair $(\hat{I}_1, \hat{I}_2)$ is derived from the first pair by injecting a static patch perturbation. Finally, Ranjan *et al.* [63] propose to compare the feature maps resulting from the predictions level-wise.

Schrodi *et al.* [68] also studied the features of FlowNetC during an attacked prediction in order to examine the reasons for its significant vulnerability to patch attacks. They identified an insufficient network architecture with a visual field that is too small together with the aperture problem as the root causes for its deficient robustness. Furthermore Schrodi *et al.* [68] complemented their analysis with an additional examination of optical flow networks against targeted global perturbation attacks. Their results indicated that given access to the weights of a network and the considered input image pair, an adversary is able to find small global perturbations that result in almost arbitrary predictions [68].

Schmalfuß *et al.* [67] concisely summarized early approaches to examine robustness of optical flow algorithms that we also want to mention here for completeness. Schmalfuß *et al.* [67] mention the study of the influence of outliers [8], noise [11], illumination changes [82] and changing the underlying dataset [1]. Furthermore they clearly distinguish the concept of adversarial robustness [63, 67, 68] from them. Furthermore Schmalfuß *et al.* [67] propose a definition for the robustness of optical flow networks that is based on an optimized constrained perturbation attack.

Carlini and Wagner [16] distinguish two approaches to evaluate robustness of neural networks generally. Adversarial attacks can be used to derive upper bounds on a networks true performance while security proofs can be used to derive lower bounds [16, 18]. As our listing above indicates the approach to study the robustness of optical flow networks using attacks seems more prevalent at the moment.

2.5 Defenses

In this work we develop attacks against the pre-processing defenses Inpainting with Laplacian Prior [3] and Local Gradients Smoothing [52]. Therefore the following section provides context about common approaches to defend optical flow networks against perturbation attacks. Intuitively, a defense modifies a neural network with the intention to increase its robustness with respect to an input domain and a performance measure [29]. Even though we focus on optical flow estimation, much of this material refers to algorithms that are originally designed for image classification. Nonetheless, they can be considered for optical flow networks.

Anand *et al.* [3] distinguish two approaches to defend optical flow networks against perturbation attacks. The first approach extends the optical flow networks with a pre-processing component that aims to repair manipulated images using transformations before they are processed. As these additions are independent of the remaining architecture, they can be applied flexibly without requiring to retrain the network [3]. Anand *et al.* [3] further distinguish pre-processing defenses that apply their transformations on the entire images on the one hand. On the other hand, local pre-processing defenses first estimate potentially manipulated regions and then apply their transformations on these estimates [3]. They mention JPEG compression [21] and Total Variance Minimization [25] as instances of global pre-processing defenses. However these algorithms are traditionally rather used in classification networks [21,25] and seem to be not thoroughly examined for optical flow estimation. On the other hand they introduce Local Gradients Smoothing [52], and Digital Watermarking [27] as examples of local pre-processing defenses [3]. In their work they also develop a local pre-processing defense that is called Inpainting with Laplacian Prior (ILP) and defends optical flow networks from adversarial patch attacks by inpainting potentially manipulated regions [3]. Additionally they analyse the performance of ILP together with the pre-processing defense Local Gradients Smoothing [52] against patches that were trained on undefended networks using the attack proposed by Ranjan *et al.* [63]. LGS is again traditionally a technique applied to image classification networks to defend them against adversarial patches [52]. Secondly Anand *et al.* [3] mention adversarial training [24] as an approach to defend networks. Its idea is to incorporate images with adversarial perturbations into the training set such that the networks learn weights that are robust against attacks [24,29,39,44,51,53].

Schrodi *et al.* [68] reason specifically for optical flow networks that an increase of the size of the effective receptive fields of a network improves its robustness to adversarial patch attacks. Consequently these authors indicate that architectural adaptations should also be considered as an approach to protect neural networks that generalizes the subclass of architectural changes introduced by pre-processing [68]. There are already many defenses published for image classification networks that introduce changes to their architecture.

Besides these distinctions there are several different approaches distinguished for image classification networks that could be considered for a transfer to optical flow networks. One potential approach introduced by Chiang *et al.* [18] is to devise defenses that yield provable security bounds on the lower robustness. There are several works that extend on this idea [86, 87]. Furthermore there are defenses distinguished that only aim to detect an attack [41, 87] and defenses that shield the networks from the attacks while guaranteeing its output response [3,52,86]. Carlini and Wagner noticed several underlying concepts that can be used to further classify defenses. They identified the concept of concept ensemble defenses [4] and mention the defenses EMPIR [71], Ensemble Diversity [55] and Error Correcting Codes [83] as instances. They also distinguished defenses that aim to introduce undifferentiable components with the intention to reduce the information usable for optimization [88].

Defenses against adversarial examples are still considered as an open problem [4, 5]. This is reflected in the ongoing development of increasingly stronger attacks and defenses in this research area [4, 14, 16]. Athalye *et al.* [5] identified an often occurring pattern that is representative for the situation. Initially a new defense is published which claims to increase a networks performance under known adversarial attacks significantly [52,58]. Subsequently, an attack is published that is specifically designed against this defense that indicates only negligible benefit of the defense [4, 15, 18]. There are several publications comprising successful attacks on various image classification defenses at once [4, 16, 18].

2.6 Guidelines for Adaptive Adversaries

There are several publications trying to formulate generalized heuristics for the successful design of robust adaptive attacks and defense algorithms [4, 14, 16]. Their aim is to sidestep the competition between attacks focused on a narrow range of defenses and defenses claiming robustness despite only testing against naive attacks [4, 14, 16].

Athalye *et. al* [4] noticed recurring patterns used by various defenses that complicate the perturbation optimization for an adversary in a white-box setting [4]. As a result these defenses often evoked the impression to be robust even though subsequent research produced successful attacks [4, 16]. Athalye *et al.* [4] summarize these patterns as obfuscated gradients and further distinguish three common subtypes. Firstly, shattered gradients completely corrupt the gradient information as a result of non-differentiable operations or numerical instabilities. Secondly, stochastic gradients introduce randomness that distorts the resulting gradient. Lastly, vanishing and exploding gradients occur in very deep networks. They stress that these properties should not be confused to actually increasing robustness and therefore propose three adapted attack schemes for overcoming every type of gradient obfuscation [4]. Finally the authors applied their new techniques successfully and devised adaptive attacks against 9 published defenses for classification networks.

Athalye *et al.* [4] propose Backward Pass Differential Approximation (BPDA) as a solution to shattered gradients. In BPDA one distinguishes between the forward and backward pass as in deep learning frameworks like PyTorch [59]. The approach then dictates to compute the forward pass normally but replace the backward pass with an approximated operation that allows the gradient information to propagate. In their experiments replacing the backward pass operation with the identity function was often sufficient [4].

We already introduced Expectation over Transformation (EoT) by Athalye *et al.* [5] in Section 2.3.4. In the context of obfuscated gradients, Athalye *et al.* [4] suggested it as a scheme to learn perturbations that are robust to randomness. More specifically, the authors proposed to approximate the actual randomized gradient of the defended network by computing the average gradient over a small batch of samples. Lastly, Athalye *et al.* [4] propose a suitable reparameterization to increase the stability in the backward pass of repeated operations for vanishing/exploding gradients.

Based on their experience from the successful development of attacks against various defenses Athalye *et al.* [4] proposed best practice guidelines that ease its independent robustness assessment by different researchers. The remainder of this paragraph summarizes their ideas that are most relevant for this work. Firstly, defenses should explicitly state a threat model. This threat model should be realistic under the intended application domain. Secondly, the robustness claim of the defense should be specific. This includes mentioning all restrictions on the perturbations and the resulting lower bound on the performance measure. Thirdly, to ease reproduction of the results, authors should release the source code and pre-trained models. Furthermore the hyperparameters associated to the robustness claims should be stated explicitly. Finally, the initial evaluation of the defense should employ an adaptive attack. Thus the attack that is used for evaluation should make explicit and significant use of information about the introduced defense [4].

However in their robustness audits against several defenses for classification networks Tramèr *et al.* [80] concluded that very often adaptive attacks used for defense evaluations are still insufficient. As a result, they published a set of recurring heuristically motivated themes that provide additional guidance for adaptive attack designers and documented their methodology [81]. We briefly summarize relevant concepts of Tramèr *et al.* [80] in the last paragraph, because they are used in the construction of our attacks.

Generally, the authors emphasize the role of an attacks simplicity. Specifically they observe that despite being tested against complex attacks there tend to exist simple attacks that significantly compromise defenses. Furthermore there are four heuristics identified that provide some guidance for the search of a suitable loss function. Firstly, the training loss should be as close to the evaluation measure as possible. As a consequence the original defense should be included if possible, modifications and additional error terms should be introduced sparsely. Secondly, the attack should aim to identify weak components of the defense and focus on them. Thirdly, the actual optimization target can have a significant impact on the success of the attack. Lastly, they stress the importance of the consistency of a loss function. Assuming one optimizes against a function that differs from the evaluation measure, then successfully optimizing for the training function should still imply success during evaluation. Furthermore a range of optimization algorithms and associated hyperparameters should be considered for the attack. Alternatives to gradient-based attacks like score-based attacks and decision-based attacks should also be considered.

3 Adversarial Patch Attacks on Defended Optical Flow Networks

This chapter introduces formal descriptions of the objects that are interacting in an adversarial patch perturbation attack on defended optical flow networks. Consequently, mathematical notations of optical flow networks, local pre-processing defenses and patch perturbation adversaries are introduced. Additionally, we provide precise algorithmic descriptions of the defense instances LGS and ILP. Furthermore the defenses are analyzed with respect to end-to-end differentiability. Subsequently, suitable modifications are presented that guarantee end-to-end differentiability. Finally, perturbation adversaries are instantiated that employ our differentiable pre-processing implementations in the construction of their loss functions.

3.1 Optical Flow Networks

We denote the set of all images by $\mathcal{I}$ and the set of all image pairs by $\mathcal{I} \times \mathcal{I}$. Within the scope of this thesis we consider the images to be encoded in RGB format with normalized entries and therefore set

$$\mathcal{I} = [0, 1]^{3 \times H \times W} . \tag{3.1}$$

A specific image instance is denoted as $I \in \mathcal{I}$ and a pair of sequential images as (I_1, I_2) with $I_1, I_2 \in \mathcal{I}$. The order of the dimensions reflects the convention in the deep learning framework PyTorch [59]. We define an optical flow network to be a function N that maps an input image pair representing sequential frames to a flow field estimation

$$N : \mathcal{I} \times \mathcal{I} \to \mathcal{F}. \tag{3.2}$$

In this equation $\mathcal{F}$ denotes the set of all legitimate flow field estimations

$$\mathcal{F} = \mathbb{R}^{2 \times H \times W}. \tag{3.3}$$

3.2 Defenses for Optical Flow Networks

The next sections introduce the local pre-processing defenses Local Gradients Smoothing published by Naseer *et al.* [52] and Inpainting with Laplacian Prior published by Anand *et al.* [3]. LGS was originally devised as a defense for image classification networks [52]. However Anand *et al.* [3] evaluated its transferability to optical flow networks in experiments together with their ILP pre-processing defense which was specifically designed to defend optical flow networks. The authors examined their influence on the prediction accuracy of a video action recognition system that employs an optical flow network as a critical component [3]. For the evaluation Anand *et al.* [3] devised an unadapted adversarial patch attack. In this adversarial setting the defenses and especially ILP yielded promising results for action recognition accuracy in the datasets UCF11 [74], KTH [69] and HMDB [38]. However they did not consider adaptive attacks in their evaluations that were optimized to overcome the respective defenses. The following sections intend to provide sufficient descriptions and analysis of the algorithms such that we are able to design our adapted attacks at the end of this chapter.

3.2.1 Pre-processing Defenses

This section introduces the model that we use to analyze the local pre-processing defenses LGS and ILP. Generally, we define a defense for an optical flow network as a functional $\mathcal{D}$, which maps an optical flow network N to a modified optical flow network $\overline{N}$:

$$\mathcal{D}(N) = \overline{N}. \tag{3.4}$$

Intuitively, this intends to reflect that a defense modifies an optical flow network in some way. The definition should be able to incorporate pre-processing and adversarial training defenses as introduced in Section 2.5. Inspired by descriptions of Anand *et al.* [3], we introduce a pre-processing defense as a special case that concatenates an independent pre-processing procedure D with the original network N:

$$\overline{N} = D \circ N, \quad \text{with} \quad D(\hat{I}_1, \hat{I}_2) = (\overline{I_1}, \overline{I_2}) \quad \text{and} \quad \hat{I}_1, \hat{I}_2, \overline{I}_1, \overline{I}_2 \in \mathcal{I}. \tag{3.5}$$

Based on this definition, the following two sections describe the local pre-processing defenses Local Gradients Smoothing [52] and Inpainting with Laplacian Prior [3] in more detail. Thus, our descriptions focus on the defenses in the context of optical flow networks. Additionally, we provide an analysis of the defenses which is aiming to detect defense components with a critical impact on end-to-end-differentiability. Based on these insights we will finally construct our adaptive patch perturbation adversaries in Section 3.3.

3.2.2 Local Gradients Smoothing (LGS)

The following descriptions are based on information provided in the original paper of LGS by Naseer *et al.* [52]. LGS protects optical flow networks against local perturbation attacks by performing two subtasks. At first, the algorithm determines potentially manipulated image regions. This is achieved by splitting the image into potentially overlapping squared blocks and a subsequent filtering operation that marks blocks with high average gradient activity. Secondly, a custom pre-processing operation that is called Gradient Smoothing is applied to the estimated regions to render them harmless.

The algorithm is based on two assumptions. Firstly, it assumes that the normalized first order gradient magnitude field of an image provides sufficient information to distinguish manipulated image regions. Secondly, it assumes that the Gradient Smoothing operation is sufficient to neutralize the malicious effect of the perturbations.

Anand *et al.* [3] transfer the defense to optical flow networks by applying its original formulation for image classification D_{LGS} on each input image separately. Because the algorithm employs four parameters this results in a parameterized family of optical flow pre-processing defenses considering our definition from Section 3.2.1:

$$D_{LGS,k,o,t,s}(I_1, I_2) = (D_{LGS}(I_1, k, o, t, s), D_{LGS}(I_2, k, o, t, s)). \tag{3.6}$$

The argument k describes the block size and o denotes the block overlap of the square block decomposition. Figure 3.1 illustrates a simplified example of a block decomposition. A block-wise filtering threshold t specifies the gradient magnitude level above which a block is considered manipulated. Lastly the smoothing parameter s determines how strong the estimated image regions are darkened depending on the actual gradient magnitude. Figure 3.2 visualizes the effect of the parameters t and s on the pre-processing result.

In the following it is assumed that the image can be seamlessly split into the overlapping blocks or otherwise they are suitably padded. Naseer *et al.* [52] did not mention a specific

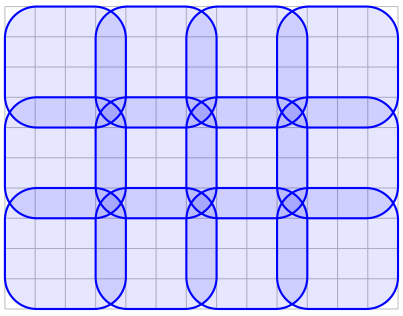


Figure 3.1: Simplified block decomposition of an image grid with height 10 pixels and width 13 pixels. The implemented blocksize is $k = 4$ and the overlap $o = 1$

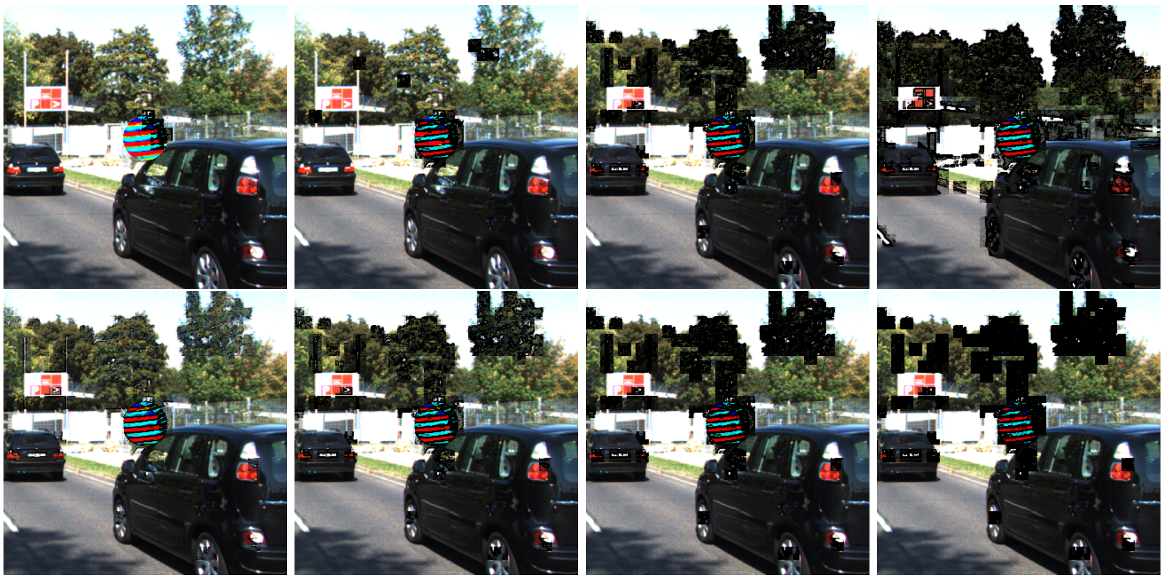


Figure 3.2: Visualization of the influence of the filtering threshold t and the smoothing factor s on a patch and its environment. The top row displays decreasing values of the filtering threshold $t \in \{0.25, 0.2, 0.15, 0.1\}$ with a fixed smoothing factor $s = 7.5$. The bottom row illustrates increasing values of the smoothing factor $s \in \{2.3, 5, 10, 20\}$ with fixed filtering threshold $t = 0.15$.

$$\begin{tikzcd}
\boxed{I} \arrow[r, "f_1(\cdot)"] \arrow[rrrrr, bend right, no head] & \boxed{T} \arrow[r, "{f_2(\cdot, k, o)}"] & \boxed{T_{unfold}} \arrow[r, "{f_3(\cdot, t)}"] & \boxed{T_{filtered}} \arrow[r, "{f_4(\cdot, k, o)}"] & \boxed{T_{fold}} \arrow[r, "{f_5(\cdot, \cdot, s)}"] & \boxed{I_{smooth}}
\end{tikzcd}$$

Figure 3.3: Data Transformation Diagram of LGS [52]. Boxes represent intermediate results and arrows visualize transformations. Visualization is inspired by a conceptually similar illustration of ILP by Anand *et al.* [3].

padding mode but we think a reflection padding is suitable to approximate the average gradient magnitude of the affected blocks. We model LGS as a procedure that applies five transformations to an input image as illustrated in the data transformation diagram presented in Figure 3.3. The next paragraphs give precise mathematical descriptions of these operations.

First, the input image I is transformed into its normalized gradient magnitude field T using the transformation f_1. In the first of three associated sub-steps, we compute the gradient magnitude fields channelwise

$$\|\nabla I(c,i,j)\| = \sqrt{\left(\frac{\partial I}{\partial y}(c,i,j)\right)^2 + \left(\frac{\partial I}{\partial x}(c,i,j)\right)^2} \qquad , \forall c \in \{r, g, b\}. \tag{3.7}$$

Naseer *et al.* [52] do not specify how to generalize their approach from grayscale images to color images. We decide to use the joint gradient as described in the lecture notes in Computer Vision by Bruhn and Weickert [12] to generalize the following steps to color images:

$$\|\nabla I(i,j)\|_J = \sqrt{\sum_{c\in\{r,g,b\}} \|\nabla I(c,i,j)\|^2}. \tag{3.8}$$

Lastly, we normalize the joint color gradient field to contain entries within the range from 0 to 1

$$f_1(I) = T \quad \text{with} \quad T(i,j) = \frac{\|I(i,j)\|_J - \min\limits_{a,b} \|I(a,b)\|_J}{\max\limits_{a,b} \|I(a,b)\|_J - \min\limits_{a,b} \|I(a,b)\|_J}. \tag{3.9}$$

Next, a block-wise decomposition f_2 is performed which yields N_v, N_h vertical resp. horizontal blocks of size k:

$$f_2(T,k,o) = F_{unfold}, \quad \text{with} \quad F_{unfold} \in \mathbb{R}^{N_v\times N_h\times k\times k}. \tag{3.10}$$

This yields the following intuitive mapping between T and T_{unfold} for a pixel at position (y,x) of the i-th vertical and j-th horizontal block, when assuming the stride of the blocks is $s = k - o$:

$$T_{unfold}(i,j,y,x) = T(s\cdot i + y, s\cdot j + x). \tag{3.11}$$

Next, we filter out blocks with an average normalized joint gradient magnitude smaller than a threshold t using transformation f_3 and receive $T_{filtered}$. Evaluating $T_{filtered}$ at the location (i,j,y,x) yields

$$T_{filtered}(i,j,y,x) = \begin{cases} T_{unfold}(i,j,y,x) & \text{if } \frac{1}{k^2}\sum\limits_{a,b} T_{unfold}(i,j,a,b) > t \\ 0 & \text{else.} \end{cases} \tag{3.12}$$

After this we recombine the blocks to get the filtered normalized joint gradient magnitude map with transformation f_4. Unfortunately, Naseer *et al.* [52] only stated to use the inverse operation of the block-wise decomposition f_2 which is ambiguous for $o > 0$. Specifically, their description is ambiguous for regions that are overlapped by multiple blocks when part of the blocks are filtered out and others not. Because we assume it is the intended formulation of the authors, we keep overlapping regions if at least one overlapping block is considered to be manipulated and thus not filtered out. Our resulting recombination transformation f_4 is based on an inofficial LGS implementation published on GitHub [50] and can be formulated as

$$f_4(T_{filtered},k,o) = T_{fold} \tag{3.13}$$

with

$$T_{fold}(i,j) = \begin{cases} T(i,j) & \text{if there is at least one unfiltered overlapping block in } T_{filtered} \\ 0 & \text{else} \end{cases} . \tag{3.14}$$

Lastly, we perform Gradients Smoothing. Intuitively, this operation darkens every pixel by multiplying the original brightness with a number from the interval $[0,1]$. This number tends to be smaller for larger s. It also tends to be smaller for larger filtered gradient magnitude values at the considered location. The Gradient Smoothing transformation f_5 is

$$f_5(I, T_{fold}, s) = I_{smooth} \quad \text{with} \quad I_{smooth}(c,i,j) = I(c,i,j) \odot \Big(\mathbf{1} - \text{clip}_{[0,1]}\{s \cdot T_{fold}(i,j)\}\Big) . \tag{3.15}$$

Naseer et al [52] mention that the parameters $k = 15, o = 5, t = 0.1$ and $s = 2.3$ worked best in their experiments, however as their source code is not provided we consider these recommendations with caution. Figure 3.3 visualizes the intermediate results of the LGS defense as well as its effect on one of our unadapted adversarial patches that we used in our experiments later in Section 4.3.1.

3.2.3 Differentiability of Local Gradients Smoothing

This section discusses the differentiability of the LGS algorithm which is relevant for the gradient-based optimization of our adversary as discussed in Section 2.3.7. Our analysis and our subsequent attack are heavily based on the brief description of Chiang *et al.* [18] about their successful attack on LGS for image classification networks. However our analysis seems to extend on their attack description as we clearly identify an optimization obstacle in the Gradient Smoothing step, that is not mentioned in their description and only occurs for strong parameter configurations. The attack of Chiang *et al.* [18] focuses on the backpropagated information of the Gradient Smoothing step described in Equation (3.15). The authors stress that despite of the non-differentiability of the other components of LGS, the Gradient Smoothing transformation provides sufficient information for a successful attack [18]. In the following we will discuss each transformation of the defense in the context of gradient obfuscation patterns by Athalye *et al.* [4] which were introduced in Section 2.6. Overall, we identify three components of the defense that can cause gradient-shattering.

Firstly the computation of the normalized joint color gradient magnitude using f_1 yields an undefined gradient at homogeneous image regions. This occurs because the derivative of the square-root function evaluated at zero is undefined. To keep the complexity of our code small we solved this issue by introducing a very small offset of size 10^{-6}. A more appropriate solution for this case would be to apply BPDA [4] to manually inject suitable gradient information at this location. Furthermore the gradient magnitude normalization could also complicate

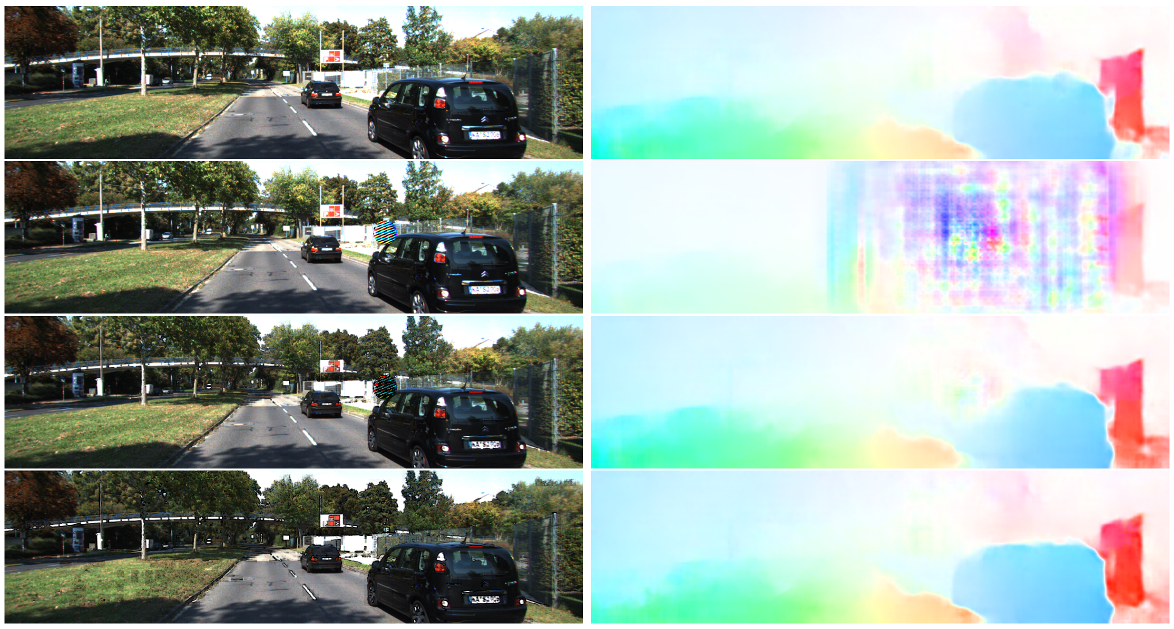


Figure 3.4: Visualization of the effect of LGS. The first column displays the first image of the image pair that served as an input to FlowNetC. The second column depicts the resulting optical flow prediction for four modalities. The modalities from top to bottom are: unattacked flow estimation without pre-processing, attacked flow without LGS pre-processing, attacked flow with LGS pre-processing, unattacked flow with LGS pre-processing. The used parameterconfiguration is $k = 15, o = 5, t = 0.1, s = 2.3$.

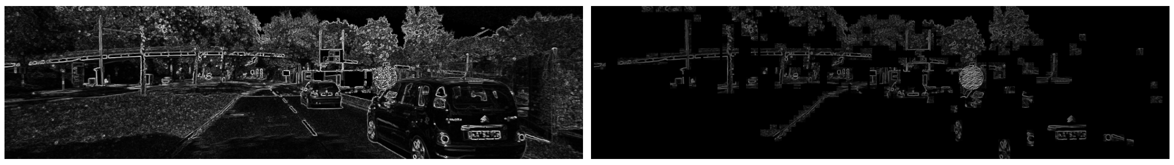


Figure 3.5: Intermediate results T (left) and T_{fold} (right) of the LGS algorithm. The used parameter configuration was $k = 15, o = 5, t = 0.1, s = 2.3$.

the parameter optimization and is additionally undefined for constant image inputs. We could not determine any evident problems with the normalization in our experiments but it could nonetheless have an undetected effect on the optimization. Additionally we neglected the undefined normalization result for constant images because it will not occur during our experiments and the issue is also not addressed in the original paper [52].

Secondly, we identify the blockwise filtering operations consisting of f_2, f_3 and f_4 as components that cause gradient shattering in our implementation as a result of the PyTorch operations that we used for the block decomposition. We restore the propagation of gradient information by applying BPDA twice and replacing the backward pass of these operations with the identity function. Specifically, the affected transformations are a division and a conditional replacement.

Finally, we identify the clipping operation during Gradient Smoothing as a critical component that causes gradient-shattering in practice for strong smoothing factors. Specifically, a lower clip in this step occurs when a pixel is darkened to a brightness value smaller than 0 and is subsequently reset to 0. As a consequence the gradient of this pixel is also reset to 0 and the previously aggregated gradient information is lost. We do not try to solve the problem of this shattered gradient in our implementation of the backwards-pass of LGS. Instead we will choose an additional error term during the construction of our Adversary against LGS in Section 4.3.2 that regularizes the perturbation again to an unclipped one. We choose this approach because it provides us more flexibility to adapt our adversary, for instance by weighting the additional error term.

3.2.4 Inpainting with Laplacian Prior (ILP)

Inpainting with Laplacian Prior is a local pre-processing defense published by Anand *et al.* [3]. It is specifically designed to protect optical flow networks against adversarial patch attacks [3]. Similarly to LGS, it performs its pre-processing separately on both input images and is again defined as a parameterized family of defenses when considering our definition from Section 3.2.1, however this time with five parameters

$$D_{ILP,k,o,t,s,d}(I_1, I_2) = (D_{ILP}(I_1, k, o, t, s, d), D_{ILP}(I_2, k, o, t, s, r). \tag{3.16}$$

The defense is conceptually based on Local Gradients Smoothing [52] but introduces design changes with two particular improvements in mind. Firstly, the authors propose to use a second order gradient statistic as an indicator for manipulated image regions instead of first order gradients [3]. Secondly they propose to use an inpainting algorithm to pre-process manipulated image regions instead of applying Gradient Smoothing on them [3].

Intuitively, the algorithm first determines manipulated image regions very similarly as Local Gradients Smoothing. Initially the second order gradient field is computed and again block-wise filtered with blocksize k, overlap o and filtering threshold t. Secondly, it the estimation is refined by filtering the currently marked regions again using the second order gradient magnitude field. However, this time the filtering is performed pixel-wise. Furthermore the second order gradient statistic is scaled by a factor s before filtering. Subsequently, a morphological operation is applied to close gaps in the estimation. Finally, the potentially manipulated regions are reconstructed using an inpainting algorithm. The following paragraphs describe the operations formally. An overview of the transformations and intermediate results is provided in Figure 3.6. The following paragraphs provide a formal description of the transformation operations in ILP.

$$\begin{tikzcd}
\boxed{I} \arrow[r, "g_1(\cdot)"] \arrow[rrrrr, bend right, no head] & \boxed{T} \arrow[r, "{g_2(\cdot,k,o,t)}"] & \boxed{T_{fold}} \arrow[r, "{g_3(\cdot,s)}"] & \boxed{M} \arrow[r, "g_4(\cdot)"] & \boxed{M_{morph}} \arrow[r, "{g_5(\cdot,\cdot,r)}"] & \boxed{I_{inp}}
\end{tikzcd}$$

Figure 3.6: Data Transformation Diagram of the Inpainting with Laplacian Prior Defense presented in a way to apply BPDA attacks.

Rather unconventionally but as introduced by Anand *et al.* [3], we define the second order gradient magnitude $\|\nabla_2 I\|$ evaluated at (c,i,j) within this work as

$$\|\nabla_2 I\|(c,i,j) = \left\| \begin{pmatrix} \partial_{yy} I(c,i,j) \\ \partial_{xx} I(c,i,j) \end{pmatrix} \right\| = \sqrt{(\partial_{yy} I(c,i,j))^2 + (\partial_{xx} I(c,i,j))^2}. \tag{3.17}$$

Again the authors did not recommend an algorithm to generalize their method to color images. We therefore again use the joint color gradient concept as described in the lecture notes of Bruhn and Weickert [12]. As a result, the second order joint gradient magnitude computation and the normalization are similar as in LGS

$$g_1(I)(i,j) = T(i,j) = \frac{\|\nabla_2 I\|_J(i,j) - \min\limits_{a,b} \|\nabla_2 I\|_J(a,b)}{\max\limits_{a,b} \|\nabla_2 I\|_J(a,b) - \min\limits_{a,b} \|\nabla_2 I\|_J(a,b)}. \tag{3.18}$$

Next the block-wise filtering g_2 is similarly defined as for LGS in Section 3.2.2. Specifically we have

$$g_2(T) = T_{fold} \quad \text{with} \quad g_2 = f_2 \circ f_3 \circ f_4. \tag{3.19}$$

We want to stress that T_{fold} in ILP contains the normalized second order joint gradient magnitude and not its first order counterpart as in LGS. After recombining the blocks, the resulting gradient field is rescaled and then filtered pixel-wise. The result is a raw inpainting mask M that marks regions to inpaint with a 1 entry

$$g_3(T_{fold}) = M \quad \text{with} \quad M(i,j) = \begin{cases} 1 & \text{if } s \cdot T_{fold}(i,j) > 0.5 \\ 0 & \text{else} \end{cases}. \tag{3.20}$$

Then, g_4 performs a morphological closing operation with a 3×3 stencil to close small gaps in the mask. The result of this operation is M_{morph}. Lastly, an inpainting algorithm is applied to recover marked regions. The parameter r specifies the radius of the environment around a pixel that is considered for its inpainting

$$g_5(I, M_{morph}, r) = \text{Inpaint}(I, M_{morph}, r). \tag{3.21}$$

The next section summarizes the Inpainting algorithm proposed by Anand *et al.* [3] in more detail. Furthermore Figure 3.7 visualizes the influence of the parameters t and s. Figure 3.8

illustrates the effect of ILP on an unadaptive patch that we will train in Section 4.3.1. Figure 3.9 shows the intermediate results of ILP.

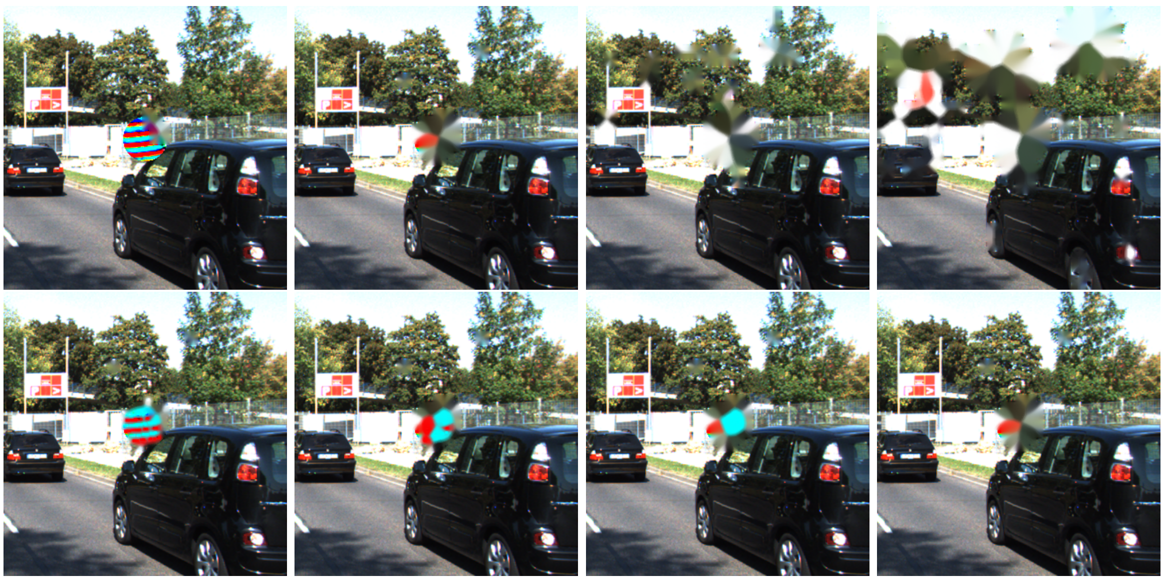


Figure 3.7: Visualization of the influence of the filtering threshold t and the scaling factor s on a patch and its environment. The top row displays decreasing values of the filtering threshold $t \in \{0.3, 0.25, 0.2, 0.15\}$ with a fixed smoothing factor $s = 10$. The bottom row illustrates increasing values of the scaling factor $s \in \{2.5, 7.5, 20, 40\}$ with fixed filtering threshold $t = 25$.

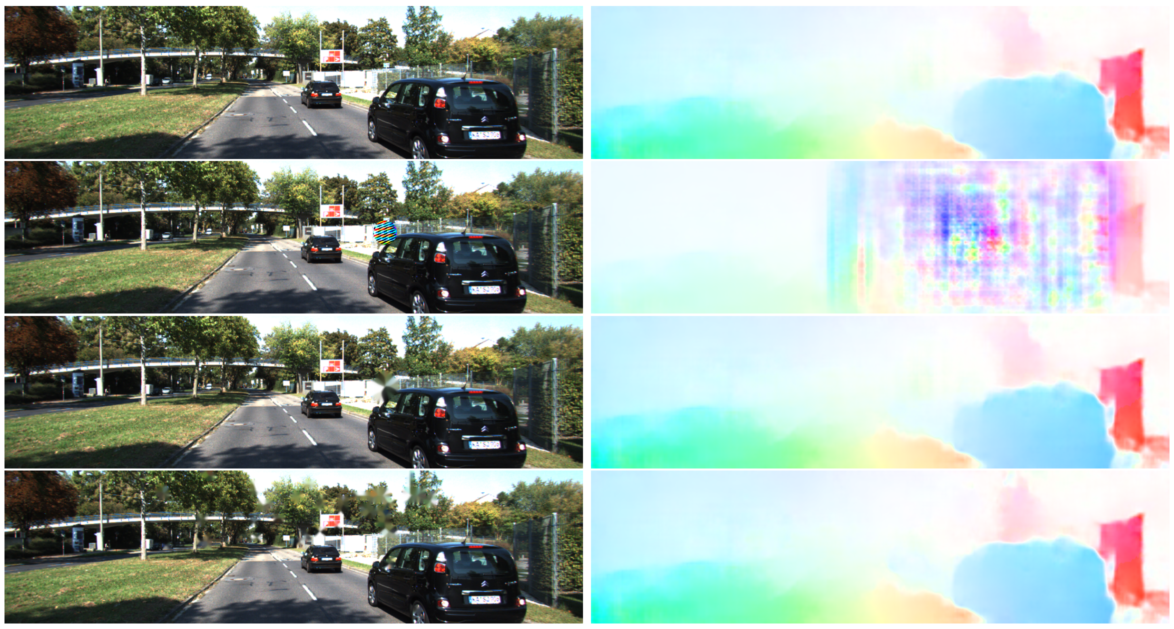


Figure 3.8: Visualization of the effect of ILP. The first column displays the first image of the image pair that served as an input to FlowNetC. The second column depicts the resulting optical flow prediction for four modalities. The modalities from top to bottom are: unattacked flow estimation without ILP pre-processing, attacked flow without ILP pre-processing, attacked flow with ILP pre-processing, unattacked flow with ILP pre-processing. The used parameter configuration is $k = 16, o = 8, t = 0.25, s = 10$.

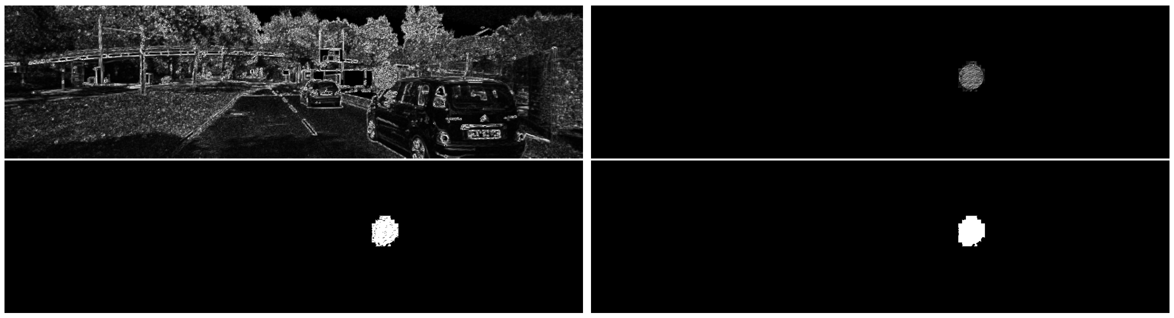


Figure 3.9: Intermediate results of the ILP algorithm. T is located at the top left. T_{fold} is situated on the top right. M can be seen at the bottom left and M_{morph} at the bottom right. The used parameters are $k = 16, o = 8, t = 0.25, s = 10$

Inpainting

Anand *et al.* [3] propose the inpainting procedure by Telea [79] to process manipulated areas. The following paragraphs present a summary of the material in the original paper [79]. The algorithm receives a RGB color image $I \in \mathbb{R}^{3\times H\times W}$ and a binary mask $M \in \{0,1\}^{H\times W}$ as an input and outputs an image $I_{inp} \in \mathbb{R}^{3\times H\times W}$ in which the marked locations are recolored using color information from the unmarked environment with a distance of less than r:

$$\text{Inpaint}(I, M, r) = I_{inp}. \tag{3.22}$$

To maximize efficiency, the algorithm aims to solve two tasks jointly. Firstly it determines an order of the marked pixels that is best suited to aid the transfer of neighborhood information into the center of marked areas. Secondly, the algorithm unmarks and computes the color of each pixel in the specified ordering sequentially.

The inpainting order is derived from the solution of the Eikonal equation that is specified by Telea [79]. As a boundary condition the pixels at the boundary B of the marked regions are set to distance 0. The remaining entries are then implied by the condition

$$|\nabla T| = 1, \text{ and } T(B) = 0. \tag{3.23}$$

Pixels with a smaller distance T to the boundary are inpainted first. The color of the pixel p is computed as a weighted average of its unmarked neighbors $\mathcal{N}_r(p)$ [79]:

$$I(p) = \frac{\sum\limits_{q\in\mathcal{N}_r(p)} w(p,q)\,[I(q) + \nabla I(q)(p-q)]}{\sum\limits_{q\in\mathcal{N}_r(p)} w(p,q)} \tag{3.24}$$

The joint weight is a product of three separate weights.

$$w(p,q) = w_{dir}(p,q) \cdot w_{dst}(p,q) \cdot w_{lev}(p,q) \tag{3.25}$$

The individual weights are defined as

$$w_{dir}(p,q) = \frac{p-q}{\|p-q\|_2} \cdot \nabla T(p) \tag{3.26}$$

$$w_{dist}(p,q) = \frac{1}{\|p-q\|_2^2} \tag{3.27}$$

$$w_{lev}(p,q) = \frac{1}{1+|T(p)-T(q)|}. \tag{3.28}$$

The first weight $w_{dir}(p,q)$ considers the angular similarity between the level-set distance map gradient ∇T at location p and the normalized distance vector from q to p. The weight $w_{dist}(p,q)$ increases the contribution of pixels q that are closer to p measured in euclidean distance. Lastly, $w_{lev}(p,q)$ increases the weight of pixels q that have a similar distance to the initial boundary as p.

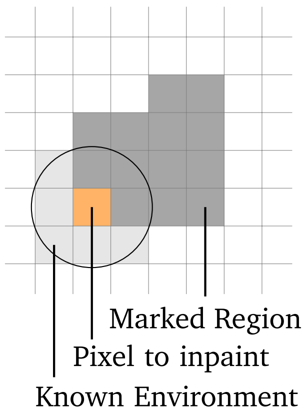


Figure 3.10: Visualization of the inpainting process of a pixel. Illustration derived from the illustration by Telea [79] for the continuous setting. The grid represents an open environment of a grid of pixels. The orange square represents a pixel that is going to be inpainted using Equation (3.24). For this inpainting operation the color values of the pixels is considered with a euclidean distance smaller than the radius r indicated by the circle. Marked pixels are not considered. As a consequence the light gray squares contribute information to the resulting brightness value of the orange pixel.

The original work of Telea specifies the algorithm with pseudocode that is structured in three procedures. We provide revised pseudocode of the original publication of Telea [79] in the appendix. In this pseudocode we corrected potential typos and added math operators that were missing in the referenced document [79]. We derived our adaptations from the textual descriptions of the original paper and from existing implementations [9, 26, 54]. Algorithm A.1 implements the backbone of the Fast Marching Method [72] that additionally performs inpainting on-the-fly [79]. It relies on two sub-procedures. The first sub-procedure Algorithm A.2 computes the resulting brightness of a pixel as the result of its inpainting as described in Equation (3.24). The second sub-procedure Algorithm A.3 specifies the update of the level-set distance map of a pixel using a computation that is described in the original work of Telea [79].

3.2.5 Differentiability of ILP

This section is based on the same premise as the analysis of LGS in Section 3.2.3. Specifically, we focus on the end-to-end-differentiablilty of the specification of the ILP pre-processing and the resulting design decisions in our implementation. Our analysis of ILP and our adaptive attack on it are based on the attacks of Chiang *et al.* [18] for LGS and Digital Watermarking [27].

Their brief attack description on LGS was summarized in Section 3.2.3. Their attack influenced our work because of the similarities between ILP and LGS with respect to the estimation of the manipulated regions. Generally, the block-wise filtering transformations of both defenses are identical. Furthermore the first and second order gradient magnitude associated statistics are very similar. Additionally, Chiang *et al.* [18] devised a successful attack against the Digital Watermarking [27] defense for image classification networks. The Digital Watermarking defense also employs the inpainting algorithm of Telea [79] to process pixels [27] and is thus related to this work. In their attack Chiang *et al.* [18] applied BPDA to the inpainting operation and approximated the backward-pass with an identity function. In the following, we transfer this idea to the ILP defense. But we first discuss general issues with respect to the differentiability of the specification.

The following backward-pass implementation of ILP aims to return gradient information only to those pixels that are not filtered out. As a result unfiltered pixels update to optimize the loss function term in which the ILP algorithm is included. Filtered pixels receive a gradient of 0. The reason why we implement the backwards path this way is that we can penalize filtered pixels with an additional error term in the loss function more flexibly as we will see in Section 3.3.3. In the following we analyze the components of ILP with respect to their differentiabilty.

Firstly, we determine the same issues with the normalized joint second order gradient magnitude transformation g_1 for ILP as for its first order counterpart f_1 in LGS which is described in Section 3.2.3. Consequently, we also introduce slight offsets to prevent a gradient shattering for homogeneous regions because of the undefined gradient of the root-function at location 0.

ILP employs the same block-wise filtering algorithm g_2 as LGS. We consequently also approximate the backward-pass with an identity function in technical contexts that are associated with the construction of an operation that is inverse to the Unfold function in PyTorch [59]. Our implementation uses PyTorch's Unfold function to realize the block-wise decomposition of the image

Thirdly, we identify the pixelwise filtering transformation g_3 as an operation that shatters gradient information. As defined in Equation (3.20) the derivative of its step function is zero almost everywhere. As a consequence it nullifies the gradient information that was gathered previously in the backward pass. However, our intended implementation does not rely on backward-pass information of the backpropagation path that is associated with the mask generation. Consequently we do not introduce any changes.

Finally, we apply BPDA [4] to the backward-pass of the inpainting procedure as proposed by Chiang *et al.* [18]. We do this, because the computations of our custom PyTorch [59] implementation are prohibitively time-consuming. As a consequence we use the optimized Python-OpenCV [9] implementation of Telea's algorithm [79]. Thus we apply BPDA to the inpainting step because it is performed using a framework that does not use an automatic backward-pass generation. Specifically, we propagate only the gradients of unfiltered pixels and set the gradients of the other pixels manually to zero. We again stress that we do this to penalize pixels that are filtered out via an external regularizing error term in the loss function. The error term intends to penalize the filtered perturbation parameters in such a way that they recover from being filtered. We give a precise description of the additional error term in Section 3.3.3

3.3 Perturbation Adversaries

In this section we instantiate three adversarial perturbation models from the building blocks examined in previous literature and summarized in Section 2.3. Our adversaries are based on the original adversarial patch attack for optical flow networks by Ranjan *et al.* [63].

Generally, we model an attack as a functional $\mathcal{A}$ that maps an original optical flow network N to a manipulated one

$$\mathcal{A}(N) = \hat{N}. \tag{3.29}$$

This allows us, inspired by the modeling of pre-processing defenses provided by Anand *et al.* [3] and the perturbation attack by Schmalfuß *et al.* [67], to describe the adversarial patch attack as a special case of an input perturbation attack. For a sequential image pair I_1, I_2 and an input perturbation adversary A_{P_1,P_2} we thus have

$$\hat{N} = A \circ N \quad \text{with} \quad A(I_1, I_2) = (\hat{I}_1, \hat{I}_2) \quad \text{and} \quad I_1, I_2, \hat{I}_1, \hat{I}_2 \in \mathcal{I}. \tag{3.30}$$

Because we develop adaptive adversaries, the following sections define three individual perturbation adversary instances. Each one is designed to attack one operation mode of FlowNetC. We consider the undefended FlowNetC network, FlowNetC with LGS pre-processing and FlowNetC with ILP pre-processing. The next section gives a detailed derivation of an adversary while the sections for the adaptive adversaries mention necessary adjustments to this baseline attack.

3.3.1 Perturbation Adversary for FlowNetC without Pre-processing

In this section we instantiate our perturbation adversary $A_{P_1,P_2} = A$ that is designed to produce adversarial patches for the undefended FlowNetC network. We parameterize our perturbations as individual RGBA tensors as described in Section 2.3.1. We set the width and height of our tensor representations equal to the image size. Thus we have $P_1, P_2 \in \mathbb{R}^{4\times H\times W}$. They are applied to the original images using alpha blending

$$A(I_1, I_2) = (\hat{I}_1, \hat{I}_2) \quad \text{with} \quad \hat{I}_i = P_{i,\alpha} \odot P_{i,rgb} + (1 - P_{i,\alpha}) \odot I_i \quad \text{for } i \in \{1, 2\} \tag{3.31}$$

We define $\odot$ to be the element-wise multiplication. $P_{i,\alpha}$ denotes the alpha channel of P_i and $\hat{P}_{i,rgb}$ denotes the rgb channels of P_i. However, to make the generated adversarial patch robust against limited rotation and scaling changes we first perform a random rotation and a random scaling on P before including the perturbation via Equation (3.31). This technique was first transferred to attacks on optical flow networks by Ranjan *et al.* [63]. The rotation angle a is randomly sampled from the interval $[-10, 10]$ and rotates P around an associated location l. Subsequently a random scaling factor b is uniformly sampled from $[0.95, 1.05]$ and the perturbation P is scaled by b originating at l.

Next, we introduce a temporal constraint as proposed by Ranjan et al. [63] to increase the practical applicability of the attack

$$P_1 = P_2 = P. \tag{3.32}$$

As a consequence we apply the same perturbation to both images which is easier to realize in practice [63]. We now overload our notation with

$$A_P(I_1, I_2) = A_{P,P}(I_1, I_2). \tag{3.33}$$

Next we introduce a constraint with respect to the possible strength of the perturbation and thus require

$$\hat{I}_i \in [0, 1]^{3\times H\times W}. \tag{3.34}$$

Because we want a circular patch of radius r we also add a spatial constraint on the perturbation considering the location of the patch to be l. We require the perturbation to only introduce changes within a radius of $r = 50$ pixels around location l. This is implemented by setting the respective alpha values of the perturbation suitably.

Tramèr *et al.* [80] recommend to use a loss function that is consistent with the evaluation goal. In our case we consider for training the flow field estimation error between the unattacked flow F and the attacked flow $\hat{F}$ as proposed by Ranjan *et al.* [63]. Additionally this yields the benefit that we do not have to adapt the ground truth flows to correct for the introduced patch [67]. Previous research about adversarial patch attacks by Ranjan *et al.* [63] yielded promising results for the ACS loss measure that was introduced in Section 2.2. To base our adversaries on firm results from the context of adversarial patches we therefore also use the ACS as a similarity measure between the training flow fields

$$\mathcal{L}(F, \hat{F}) = ACS(F, \hat{F}). \tag{3.35}$$

Finally we optimize our perturbation P using Projected Gradient Descent on the allowed perturbation interval $[0, 1]$ with updates clipped to the range $[-2/255, 2/255]$ and a learning rate of 1000.

3.3.2 Adversarial Model for FlowNetC with LGS Pre-processing

In the following we state the adjustments to our baseline adversary from the previous chapter, that are necessary to receive our adaptive adversarial patch attack on LGS. Overall we apply two modifications to our initial adversary.

Firstly, we compute the ACS loss between the unattacked but LGS pre-processed flow estimation F_{LGS} and the attacked and defended flow estimation $\overline{F}_{LGS}$. We stress that this implies the inclusion of our LGS implementation into our training loss. Thus the implications summarized in Section 3.2.3 take effect. However for very strong smoothing factors there occurs the situation where the gradient information of the ACS is zero for parameters that are clipped via gradient smoothing. This is a result of a large normalized joint gradient magnitude of the affected perturbation parameter together with a strong filtering threshold. To respond to this and to restore the gradient information of the ACS loss term, we add another error term that penalizes the average joint gradient magnitude of the patch. Let $\|\nabla P\|_J$ denote the joint gradient magnitude field of the patch P. Let furthermore n denote the total number of pixels that are perturbed by P. Then we can formulate our loss function with a weighting coefficient α as

$$\mathcal{L}_{LGS}(F_{LGS}, \overline{F}_{LGS}, P) = ACS(F_{LGS}, \overline{F}_{LGS}) + \alpha \frac{\sum\limits_{i,j} \|\nabla P\|_J(i,j)}{n}. \tag{3.36}$$

We optimize our LGS adversary generally using I-FGSM [39] updates because this allows us to choose the coefficient α of the regularization term negligible small for pixels with existing gradient information. Our perturbation parameters are additionally clipped to the range $[0, 1]$ after every update step. Generally, we choose the I-FGSM update size to be $\epsilon = 0.004$ and set α to 10^{-8}. We will see in Section 4.3.2 that there will be one adversary that we will adapt from this schedule to increase the quality of its trained perturbation result.

3.3.3 Adversarial Model Designed for FlowNetC with ILP Pre-processing

This section summarizes the modifications to the baseline adversary from Section 3.3.1 that are necessary to receive our attack on ILP. Overall, we introduce adjustments to the loss function that are analogous to the changes for our LGS adversary in Section 3.3.2.

Firstly, our adaptive adversary trains its perturbation on an flow prediction error term that incorporates the ILP defense. Specifically, our ILP adversary minimizes the ACS loss between the unattacked but ILP-defended flow prediction F_{ILP} and the attacked and ILP-defended flow prediction $\overline{F}_{ILP}$. This adjustment is central, as it provides our adversary with the information that is necessary to learn a perturbation with a strong impact on the prediction despite being pre-processed by ILP.

To provide filtered pixels with gradient information we introduce an additional error term that penalizes the patches average second order joint gradient magnitude statistic $\|\nabla_2 P\|_J$ as defined in Section 3.2.4. The resulting loss function that is weighting the regularization term with weight α is

$$\mathcal{L}(F'_{ILP}, \overline{F}_{ILP}, P) = ACS(F'_{ILP}, \overline{F}_{ILP}) + \alpha \frac{\sum_{i,j} \|\nabla_2 P\|_J(i,j)}{n}. \tag{3.37}$$

Similarly to the previous attack the regularization term penalizes the second order joint gradient magnitude of inpainted perturbation parameters in order to help them from being marked during the inpainting mask generation of ILP. Because we again use the I-FGSM optimization approach and a very small coefficient $\alpha = 10^{-8}$, our regularization term effectively penalizes only the perturbation pixels that are detected during the mask generation. This is because we implemented the backward-pass of ILP in such a way that these perturbation parameters receive a zero gradient as gradient information from the ACS term. We however clip the resulting parameters after every update to the range $[0, 1]$ and select an update size of $\epsilon = 0.004$.

4 Experiments

To evaluate the efficiency of our adversarial models, we conducted experiments consisting of a training run and a subsequent evaluation. This chapter summarizes the results of our attacks when they are applied to the FlowNetC network with resp. without pre-processing defenses. Our experiments use the train/test split proposed by Ranjan *et al.* [63]. We train our adversaries on the KITTI Raw dataset [23] and test on the training subset of the KITTI15 dataset [48]. The remainder of this chapter is structured as follows.

The next section provides information about our employed software. In the third section we describe our training and evaluation procedures. The fourth section reports our experimental results for the undefended, LGS-defended and ILP-defended FlowNetC. Finally, the fifth section discusses our results.

4.1 Software

We implemented our experiments in PyTorch [59]. Furthermore, we employ the pretrained FlowNetC model and class from the GitHub repository associated with the original adversarial patch attack by Ranjan *et al.* [61, 63]. From this repository we also use the image loader functions for the KITTI Raw dataset [61]. We use the data loader functions from the RAFT repository [2] to load the KITTI15 training dataset. Additionally we used the PyTorch Correlation Module for FlowNetC by ClementPinard [60]. Our re-implementation of LGS is strongly influenced by a LGS implementation for image classification networks published on GitHub [50]. For the inpainting algorithm we used the Python-OpenCV implementation of Telea's algorithm [9] For the inpainting mask generation in ILP we again strongly considered the LGS implementation mentioned previously [50]. Our Training and Evaluation procedures are derived from the original adversarial patch attack repository by Ranjan *et al.* [63]. We created diagrams with the graphing library Plotly [31] and flow field colorplots using the flow_library GitHub repository by Mehl [47].

4.2 Training and Evaluation Procedure

Algorithm 4.1 summarizes our training procedure. Let F_D denote an unattacked flow field estimation that applies the pre-processing function D. This pre-processing function can also be the identity function to include the undefended network in our statements. Let furthermore $\overline{F}_D$ denote an attacked flow field prediction which applies the pre-processing D. Then, our training procedure iteratively updates our perturbation parameters P from randomly sampled image pairs by minimizing the adversarial loss function $\mathcal{L}$ that depends on the unattacked and attacked prediction $F_D, \overline{F}_D$ as well as the perturbation parameters P. In the computation of our ACS loss we ignore the image region that is occluded by the circular patch.

Each sampled image pair is rescaled by independent horizontal and vertical factors from the interval $[1, 1.5]$ and subsequently cropped to a height of 320 and a width of 512 to increase the computation speed. After computation of the adversarial loss function $\mathcal{L}$ backpropagation is used to generate gradient-information to update our perturbation parameters. At the end of an iteration, our adversary updates its parameters using the respective optimization update and learning rates. To improve reproducibility we initialize the pseudo-random number generator with a fixed seed at the beginning of every training run.

Algorithm 4.1 Training procedure for a perturbation model $A_P = A$ on the optical flow network FlowNetC denoted as N. The experiment uses the KITTI Raw dataset which is denoted as $DS_{KITTIRaw}$ and considers a defense $D \in \{D_{\text{LGS},k=15,o=5,t,s}, D_{\text{ILP},k=16,o=8,t,s,r=5}, Id\}$. The adversary instance applies its perturbation to the image as described in Section 3.3. The update procedure U is dependent on the adversarial model. The learning rate is α. Gradient-information is computed via PyTorch's [59] autograd functionality.

```
function TRAIN($A, N, DS_{KITTIRaw}, D, U, \alpha$)
    for $i \in \{1, \dots, n\}$ do
        Sample $(I_1, I_2)$ uniformly from Dataset $DS_{KITTIRaw}$
        Apply RandomRescalingThenCrop to $I_1, I_2$
        $\overline{I_1}, \overline{I_2} := D(A_P(I_1, I_2))$                // Attack and Pre-process
        $I'_1, I'_2 := D(I_1, I_2)$                                          // Pre-process
        $\overline{F}_D, F_D := N(\overline{I_1}, \overline{I_2}), N(I'_1, I'_2)$   // Estimate Flow Fields
        $L := \mathcal{L}(\overline{F}_D, F_D, P)$                           // Compute Error
        $P = U(P, \frac{\partial L}{\partial P}, \alpha)$                    // Update Patch
    end for
    return A
end function
```

Our evaluation experiment is built upon the evaluation experiments of previous attacks on neural networks [63, 67, 68]. We compute the AEE between the unattacked prediction F_D and the attacked prediction $\overline{F}_D$ on the one hand to estimate the network's resulting robustness [67]. On the other hand, we compute the AEE between the ground truth flow and the attacked flow $\overline{F}_D$ to estimate the networks resulting prediction accuracy [67]. In both cases we ignore the regions that are occluded by the inclusion of the patch. Algorithm 4.2 summarizes our evaluation procedure. In our evaluation we iteratively compute the AEE of the images contained in the KITTI15 training split. In one iteration we first apply a center crop to the image to guarantee that its height and width are the largest possible multiples of 64 which is required by FlowNetC [61]. Then, we apply the pre-processing D on the unattacked image pair and on an attacked version of it. After this we estimate the attacked and unattacked flows $\overline{F}_D, F_D$. In the end of an iteration, we compute $AEE(F_{gt}, \overline{F}_D)$ and $AEE(F_D, \overline{F}_D)$. Finally we return the average of the respective AEE computations. Again our code initially resets the pseudo-random number generators to increase reproducibility.

Algorithm 4.2 Evaluation procedure for a perturbation model attacking an undefended network N. The adversary A_P is tested against the optical flow network N on the KITTI 2015 training dataset $DS_{KITTI15}$ [48]. We stress $DS_{KITTI15} \cap DS_{KITTIRaw} = \emptyset$. The center crop reduces the images' size to $H = 320$ and $W = 1216$ as this seems to be the largest multiples of 64 that are contained in every frame of $DS_{KITTI15}$. We assume L and L_{gt} are initialized to zero tensors. We assume $D \in \{Id, D_{LGS,15,5,t,s}, D_{ILP,16,8,t,s}\}$.

```
function EVALUATE($A_P, N, D, DS_{KITTI15}$)
    for $(I_1, I_2, F_{gt}) \in DS_{KITTI15}$ do
        Apply CenterCrop to $I_1, I_2, F_{gt}$
        $(I'_1, I'_2) := D(I_1, I_2)$
        $(\overline{I}_1, \overline{I}_2) := D(A_P(I_1, I_2))$
        $F_D, \overline{F}_D := N(I'_1, I'_2), N(\overline{I}_1, \overline{I}_2)$
        $L_{gt}$ += $AEE(F_{gt}, \overline{F}_D)$
        $L$ += $AEE(F_D, \overline{F}_D)$
    end for
    return $\frac{L_{gt}}{n}, \frac{L}{n}$
end function
```

4.3 Evaluation Results

The following sections present the evaluation results for our adversaries against the FlowNetC network without pre-processing and with LGS resp. ILP pre-processing. The attacks on the undefended FlowNetC serve as a baseline to determine the network's vulnerability. Additionally we test the performance of selected patches against a range of parameter configurations of t and s for LGS and ILP. Based on the performance of our baseline patches, we select challenging parameter-configuration instances t, s for LGS and ILP to evaluate our adaptive attacks against them.

4.3.1 Attacks on the Undefended FlowNetC Network

In the first experiment we train five different instances of the adversarial model defined in Section 3.3.1. We summarize the most important adjustments. We train our randomly placed patches against the $ACS(F, \hat{F})$ loss function between the attacked and unattacked flow $F, \hat{F}$ without any pre-processing. Our adversary performs projected gradient descent updates that are clipped to size $1/255$ and use a learning rate of $1e3$. We initialized our RGBA patch to have a non trainable circular alpha layer with a diameter of 50 pixels and set all RGB values to 0 initially. We then train our patches for $10,000$ iterations.

Figure 4.1 shows the moving average of width 100 of the training loss. We notice that the patches seem to be sufficiently converged after approximately 4000 iterations. Additionally we see that the curves of run three and four converge to a different loss value than the remaining runs. This can be an indicator that there might be several local minima that are dominating the parameter space. The visual appearance of the trained patches is illustrated in Figure 4.2. In these runs, there seem to be two patterns dominating, that can be distinguished by their frequencies and color distributions. Run 3 and run 5 seem to be very pure representations of these patterns. Further comparison with the training loss function indicates that the patches with the higher frequency pattern are Run 3 and Run 4 that converged to a limit that is less efficient as the other three patches. We included a visualization of the evolution of the patches over the first 7000 iterations in the appendix Figure A.1.

For comparison we also trained five patches using the official repository of the adversarial patch attack by Ranjan *et al.* [61] with standard parameters. Figure 4.3 visualizes the resulting patch

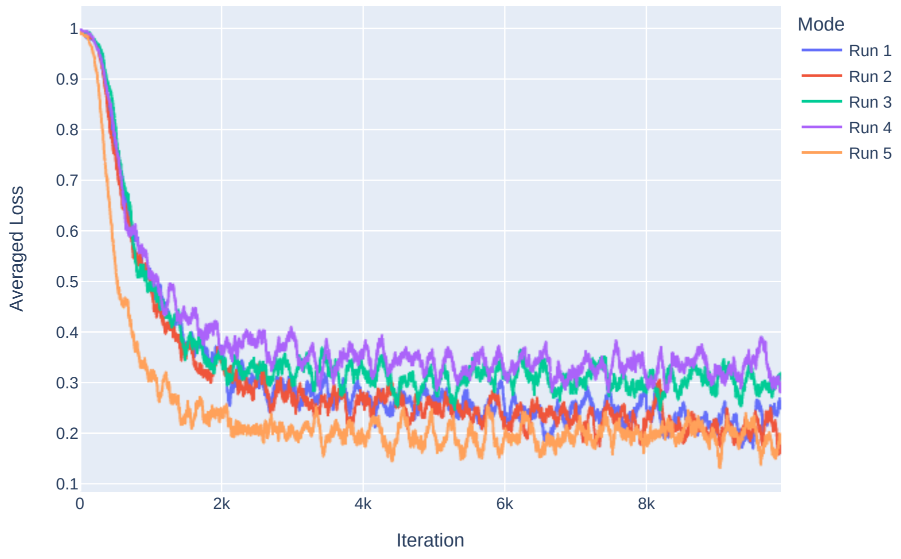


Figure 4.1: Moving average of kernel size 100 of the training losses over 10000 training iterations for 5 similarly initialized runs.

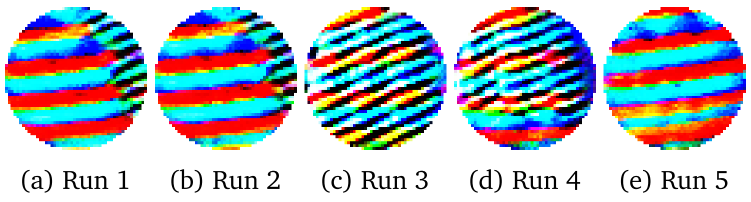

(a) Run 1 (b) Run 2 (c) Run 3 (d) Run 4 (e) Run 5

Figure 4.2: Adversarial patches resulting from five instances of our adversary against the undefended FlowNetC as described in Section 4.3.1. The adversary instances are trained using our custom training procedure described in Algorithm 4.1.

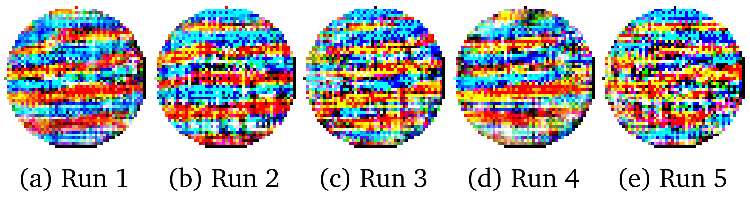

(a) Run 1 (b) Run 2 (c) Run 3 (d) Run 4 (e) Run 5

Figure 4.3: Adversarial patches resulting from five executions of the training procedure of the adversarial patch attack repository by Ranjan *et al.* [61] with standard parameters. The visualizations are derived by clipping the actual adversarial patch tensor to the range $[0, 1]$.

perturbations. We notice that these patches are dominated by noise like brightness fluctuations. However we can also clearly see that the general appearance of the patches resembles the appearance of our custom patch from run 5 that is shown in Figure 4.2e. This pattern appears as horizontal stripes in the colors blue and red.

The code of the flowattack GitHub repository of Ranjan *et al.* [61] however seems to contain some inconsistencies that we explain in the following. Initially, we observe that the original tensor representation of a trained RGB perturbation contains entries exceeding the range $[-3, 3]$. Further analysis of the update procedure indicates that the perturbation parameters are not optimized as a constrained optimization problem. Instead the patch is optimized as a global optimization problem and the rgb perturbation parameters are clipped every time before they are injected into the images. Specifically, the patch is clipped to the interval $[0, 1]$ which is also the brightness range of the images. Furthermore the training procedure does not update the perturbation parameters using the direct gradient $\frac{\partial L}{\partial P}$. Instead the update is derived from the gradient of the attacked images $\frac{\partial L}{\partial \hat{I}}$ at the location of the patch. As a consequence, the patches trained with the flowattack repository by Ranjan *et al.* [61] grow out of the range $[0, 1]$ even though the clipping operation would normally shatter the gradient for clipped pixels. We assume that these inconsistencies are the reason for the visual difference between our custom patches and the patches by Ranjan *et al.* [63].

Custom Training Pipeline	Unattacked	Run 1	Run 2	Run 3	Run 4	Run 5
$AEE(\hat{F}, F)$	0.0	27.020	28.011	17.944	15.901	36.919
$AEE(\hat{F}, F_{gt})$	12.997	37.283	38.201	28.722	26.893	46.645

Ranjan *et al.* Repository	Unattacked	Run 1	Run 2	Run 3	Run 4	Run 5
$AEE(\hat{F}, F)$	0.0	10.123	7.147	6.337	9.563	5.121
$AEE(\hat{F}, F_{gt})$	12.997	21.433	18.801	18.113	20.906	17.076
$AEE(\hat{F}, F_{gt})$ (test_patch.py)	14.515	24.654	22.041	21.352	24.067	19.765

Table 4.1: Evaluation results of Algorithm 4.2 for the patches trained with our procedure in the upper table and patches trained with the flowattack GitHub repository by Ranjan *et al.* [61]. For the patches trained with the flowattack repository we also included the AEE scores returned by the associated function test_patch.py [61].

Table 4.1 summarizes the performance of the patches generated with our training pipeline and the patches trained using the repository of Ranjan *et al.* [63]. The table is separated in two parts. The upper part shows the evaluation results of our patches and the lower table presents the evaluation results for the patches trained with the flowattack repository [61]. Generally, we used Algorithm 4.2 for the computation of $AEE(\hat{F}, F)$ and $AEE(\hat{F}, F_{gt})$. Because of the minor performance of the patches trained by the Flowattack repository, we included the results of the evaluation procedure that is provided with the Flowattack repository. We can reproduce the reported evaluation results of an AEE of more than 40 pixels for a patch of diameter 50 pixels which was determined by Ranjan *et al.* [63]. Our Run 5 produced an AEE of more than 46 pixels between the prediction and ground truth flow in our tests.

Additionally we notice that our patches which consist mainly of the lower frequency pattern (Run 1,2 and 5) result in a larger AEE compared to patches that consist of the higher frequency pattern. Furthermore the patches that we trained using the repository of Ranjan *et al.* [61] performed consistently worse than our custom patches. We assume the reasons for this performance gap are the inconsistencies in their training procedure described above.

Figure 4.4 visualizes the effect of our patches that were trained using Algorithm A.1 on the prediction result of FlowNetC. All patches have a dominating impact on large areas around their location. This is consistent with our evaluation results that yield very large AEEs for the error measures between the unattacked and attacked flows. Furthermore all introduced flow field distortions appear very similar.

Finally we estimate the prediction accuracy of FlowNetC with LGS resp. ILP pre-processing when attacked by the best of the patches that we trained in our training pipeline and the best of the patches that we trained using the repository of Ranjan *et al.* [61]. Figure 4.5 summarizes their performances by computing the AEE between the flow field ground truth F_{gt} and the attacked and pre-processed prediction $\overline{F}$ for a wide range of parameter configurations of t and s. Specifically, we tested the defenses with their recommended blocksize k and overlap o and varied the parameters s and t jointly within the ranges $s \in \{0, 3, \dots, 27, 30\}$ and $t \in \{0, 0.04, \dots, 0.36, 0.4\}$ to generate the interpolated heatmaps displayed in Figure 4.5. At this point we want to stress that the smoothing factor of LGS and the pixel-wise scaling factor in ILP represent different concepts and should not be compared directly as suggested by the plots. In the following paragraphs we first describe an intuitive interpretation of the heatmaps and analyze the results subsequently.

Intuitively the heatmaps for LGS in Figure 4.5a and Figure 4.5c can be interpreted as follows. For the following interpretation consider the heatmap presented in Figure 4.5a. Increasing the filtering threshold t decreases the defense's sensitivity to gradient changes. Consequently,

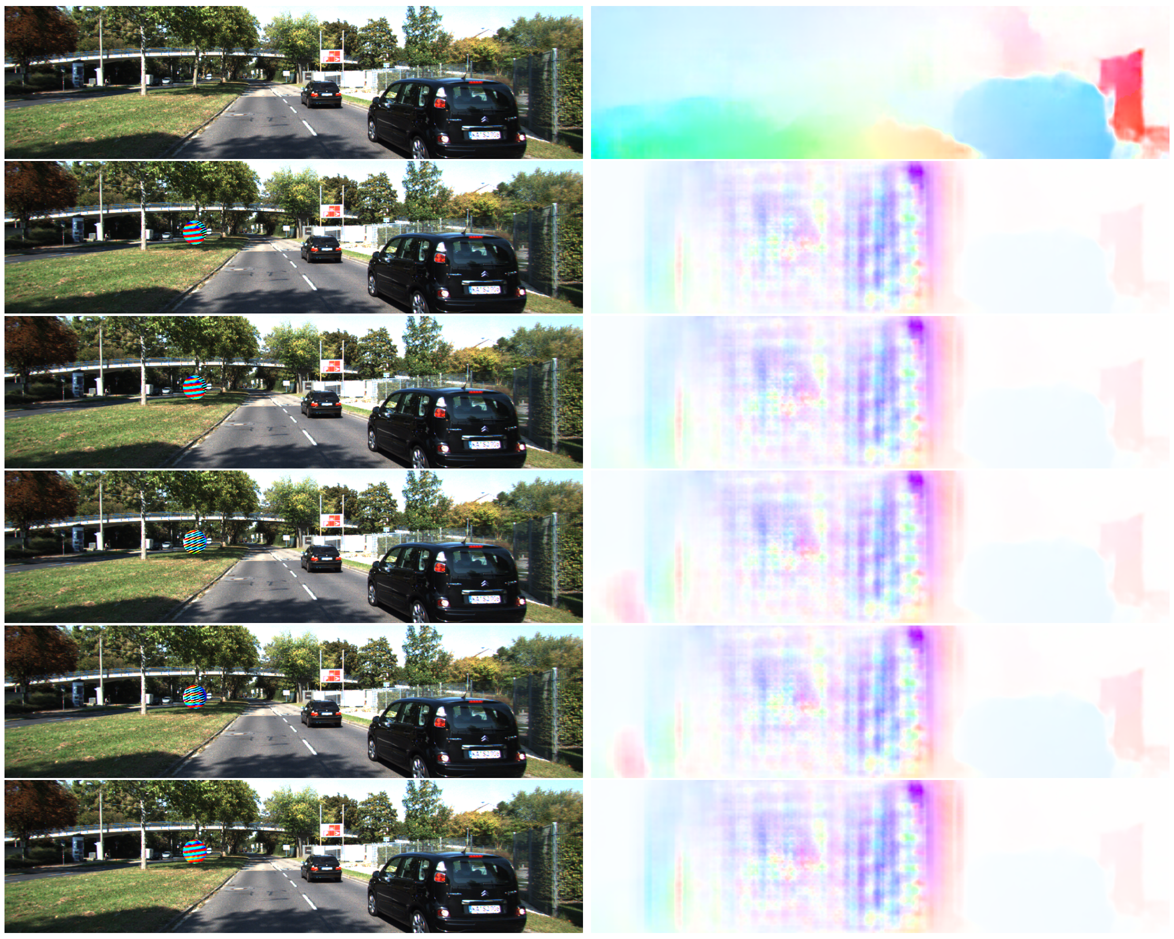
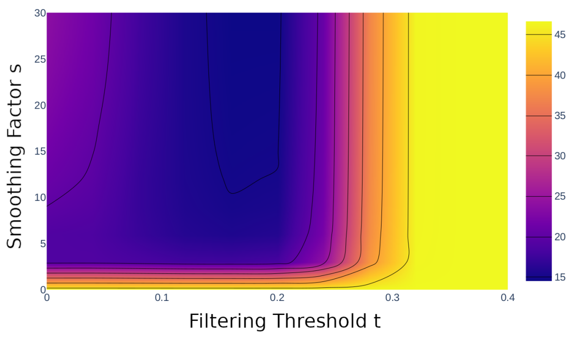
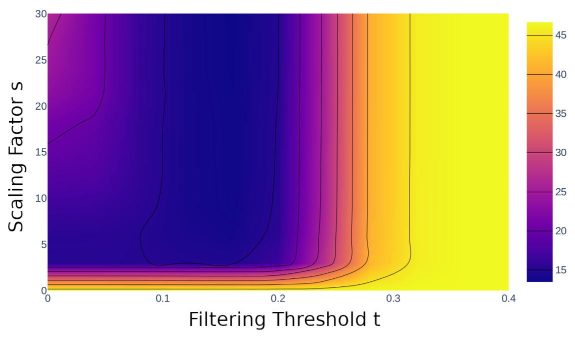


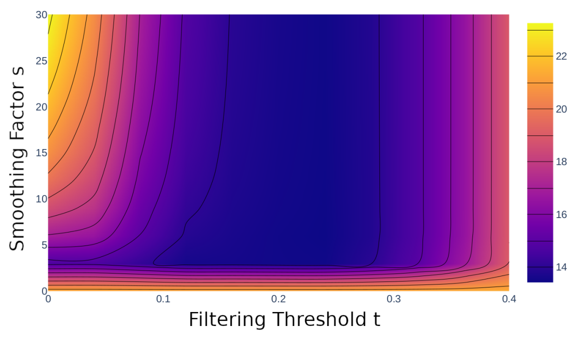
Figure 4.4: Influence of our adversarial patches trained with Algorithm 4.1 on the FlowNetC prediction result. The first row visualizes an unattacked input image together with the flow field estimation. The following rows present the results for images that are attacked by our patches of Run 1 to Run 5.



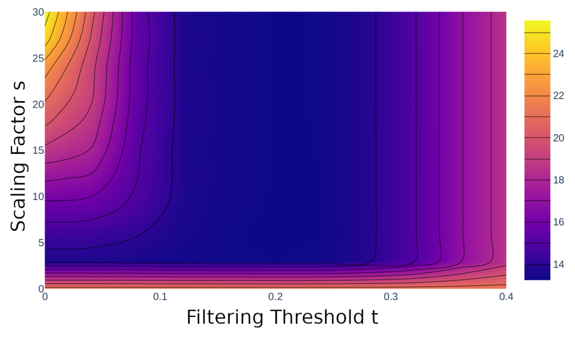
(a) $AEE(F_{gt}, \overline{F}_{LGS})$ for the patch from run 1 trained using our custom training pipeline.



(b) $AEE(F_{gt}, \overline{F}_{ILP})$ for the patch from run 1 trained using our custom training pipeline.



(c) $AEE(F_{gt}, \overline{F}_{LGS})$ for the patch from run 1 trained using the repository by Ranjan *et al.* [61]



(d) $AEE(F_{gt}, \overline{F}_{ILP})$ for the patch from run 1 trained using the repository by Ranjan *et al.* [61]

Figure 4.5: Heatmaps displaying AEE between the flow field ground truth F_{gt} and the attacked flow and pre-processed prediction $\overline{F}_D$ with defense D. over the parameter space of the filtering threshold t and the smoothing/scaling factor s for the defenses when evaluated using Algorithm 4.2 on KITTI 2015 (Training).

the yellow region on the right side displays parameter configurations that apply smoothing to almost no region. On the other side, decreasing the smoothing factor s reduces the effect of the smoothing operation. As a result the yellow regions on the bottom show parameter configurations in which the Gradient Smoothing operation has almost no effect. The lighter area in the top left corner represents parameter conigurations that introduce a bias error as a result of excessive Gradient Smoothing in almost all areas of the image. Lastly, there is a vertical purple stripe representing efficient parameter configurations of t and s. The interpretation of the heatmap of ILP is similar.

Both algorithms are able to correct the attacked flow prediction such that the AEE to the ground truth flow decreases below 15 pixels. We estimated the $AEE(F_{gt}, F)$ of FlowNetC to be approximately 13 pixel as displayed in Table 4.1. Consequently there are defense parameterizations that reduce the $AEE(\overline{F}, F_{gt})$ from AEEs of more than 45 resp. 22 down to below 15 resp. 14. The contour line indicating regions with an AEE of less than 15 pixel is less restrictive for ILP than for LGS in the heatmaps Figures 4.5b and 4.5d. This supports the claims of Anand *et al.* [3] that their second order gradient statistic is a better estimator of manipulated image regions. Additionally we see that the recommended parameter configuration for ILP of $t = 0.25$ and $s = 10$ by Anand *et al.* [3] performs well for the patch trained with the flowattack repository in Figure 4.5d but not well for our patch in Figure 4.5b.

We conclude that both defenses indeed increase the network's robustness against unadaptive adversarial patch attacks. Our goal in subsequent experiments is to derive adversaries that introduce significantly more robust perturbations by including information about the defense into our training procedure. For this we train our adaptive adversaries against selected parameter configurations that yielded promising results in Figure 4.5a for LGS and Figure 4.5b for ILP.

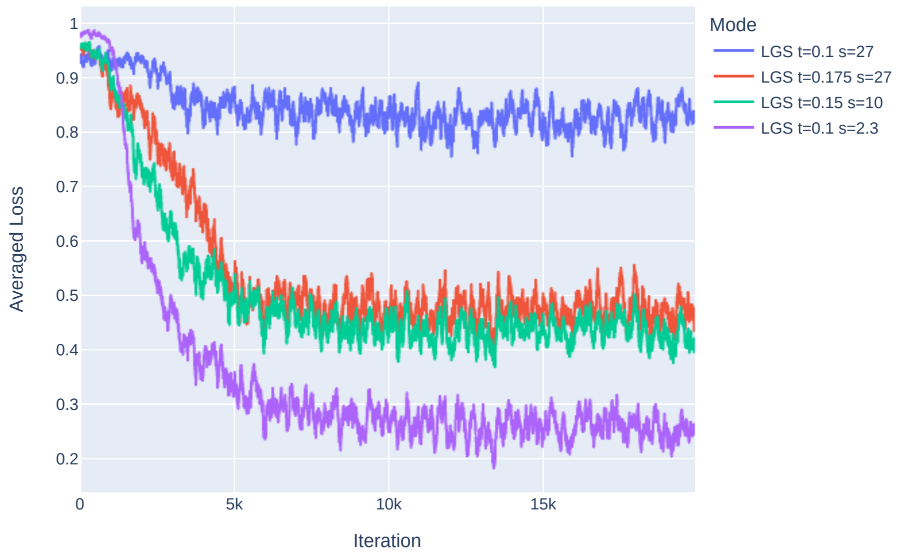
4.3.2 Attacks on FlowNetC with LGS Pre-processing

To empirically evaluate the effectiveness of our adversaries against LGS we selected two parameter configurations of the filtering threshold t and the smoothing factor s that performed well in our experiments on unadapted attacks summarized in Figure 4.5a. This resulted in the parameters $(t, s) \in \{(0.175, 27), (0.15, 10)\}$. Furthermore we selected with $(t = 0.1, s = 27)$ one parameter configuration that is very restrictive to test the convergence behavior. Additionally we tested the parameter configuration that performed best for the defense authors [52] and thus also added $(t = 0.1, s = 2.3)$.

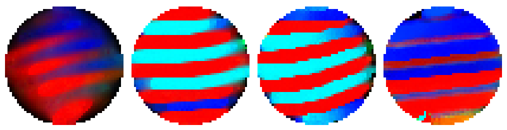
Initially we want to stress that we adapted the optimization algorithm of our adversarial model targeting the parameter configuration $(t = 0.1, s = 2.3)$ because it converged to a suboptimal patch for the baseline configuration against LGS. Specifically we changed the optimization update to a clipped Projected Gradient Descent update and set the loss function coefficient $\alpha = 0.001$. We set the maximum update size of Clipped PGD to 0.008 and set the learning rate to 10.

Figure 4.6a shows the moving average of size 100 of the training loss of the four runs over 20,000 iterations. We clearly see that the adversary that is training against the most restrictive parameter configuration of $t = 0.1, s = 27$ has problems to deflect the flow. The remaining patches converge to more successful training values of the ACS. Especially the patch that is trained against the parameters $t = 0.1, s = 2.3$ achieves a promising training score. We stress at this point that the patches train against different loss functions as each of them trains against a different instantiation of the LGS defense. This has to be considered when one compares the training losses.

Additionally Figure 4.6b presents the resulting patches. The patches have evident differences compared to the patches trained in Section 4.3.1. We however notice that the pattern of blue and red horizontal stripes is still visible in all our patches. Additionally the patches evolved



(a) Moving average of kernel of size 100 over the training losses of our attacks on FlowNetC with LGS pre-processing.



(b) The resulting patches of our training procedure when training on FlowNetC with LGS and parameters $k = 15, o = 5$ and from left to right $(t, s) \in \{(0.1, 27), (0.175, 27), (0.15, 10), (0.1, 2.3)\}$.

	$t = 0.1,$ $s = 27$	$t = 0.175,$ $s = 27$	$t = 0.15,$ $s = 10$	$t = 0.1$ $s = 2.3$
$AEE(\overline{F}, F')$	4.118	12.490	12.427	27.531
$AEE(\overline{F}, F_{gt})$	18.231	23.504	22.952	35.625

Figure 4.6: Evaluation results when applying the evaluation procedure specified in Algorithm 4.2 on the trained patches.

sharper edges and are dominated by an increased degree of homogeneity. Furthermore we see that the patches of run 2 and run 3 appear visually similar.

Figure 4.6 displays the evaluation results of our patches using Algorithm 4.2. Generally, all patches increase the AEE by at least some pixels. The patch trained with a strong filtering threshold $t = 0.1$ but a small smoothing factor $s = 2.3$ performed best. On the other hand the patch trained with the same filtering threshold $t = 0.1$ but a very strong smoothing factor $s = 27$ performed worst. Our most successful run was able to induce an AEE between the attacked and unattacked flow of more than 27 pixels. On the other hand, our worst performing patch resulted in an equivalent AEE of more than 4 pixels. Consequently our patches seem to have increasingly more difficulties to manipulate the flow prediction for increasingly restrictive parameters.

Figure 4.7 visualizes the effect of the patches on the defended flow predictions. When comparing the unattacked flow field prediction in row 1 with the attacked flow field predictions in the remaining rows, we clearly see that the second, third and fourth run are able to distort the defended flow estimations significantly. On the other hand the first run with very restrictive parameters has only a smaller effect on the defended flow prediction. However

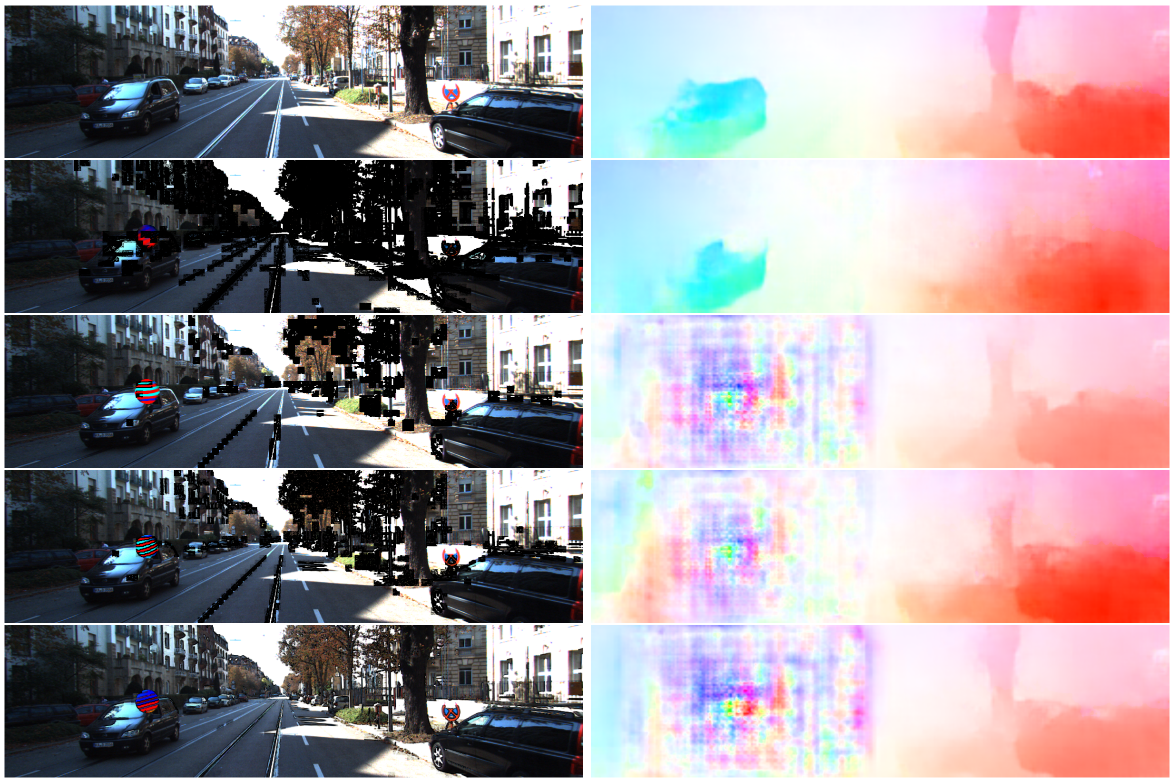


Figure 4.7: Comparison of the effect of the individual patches trained in experiment 2 on the flow field prediction.

when considering the pre-processing results in the first column of Figure 4.7 one also sees that the LGS pre-processing with $t = 0.1, s = 27$ darkened large areas of the image significantly. Consequently, training for this configuration seems to be more difficult.

4.3.3 Attacks on FlowNetC with ILP Preprocessing

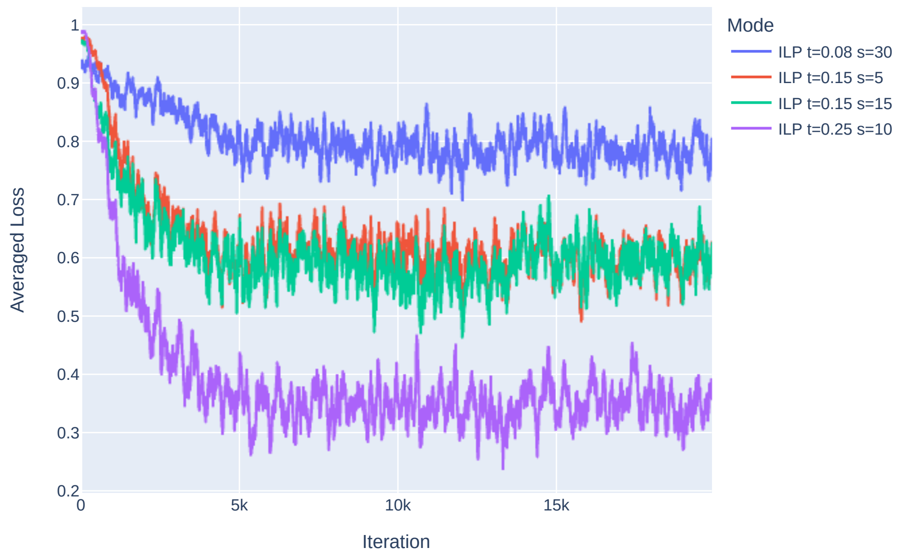
In this section we summarize the evaluation results of our attacks on FlowNetC with ILP pre-processing. The adversarial model that generates and applies our adversarial patches was introduced in Section 3.3.3. Overall we perform four experiments with different parameters s and t for ILP. The parameters $t = 0.25, s = 10$ were recommended by the authors of the ILP defense [3]. Additionally, we selected two parameters for our experiments that performed well against our strongest adversarial patch that was trained in Section 4.3.1. Specifically we considered the contour plot in Figure 4.5b and selected two parameters that are situated central in promising regions of the parameter space. We selected the parameters $t = 0.15, s = 5$ and $t = 0.15, s = 15$ from this figure. Additionally we selected the very restrictive parameter combination $t = 0.08, s = 30$ to test the convergence behavior of the adversary.

Figure 4.8a shows an moving average of size 100 of the training losses over time. One can clearly see that the patches converge towards different limits. The parameters $t = 0.15, s = 5$ and $t = 0.15, s = 15$ seem to converge to related limits. Generally the curves indicate that most patches successfully converged to minima. However especially the patch trained with $t = 0.08, s = 30$ converged to a relatively weak training loss. Also the training loss of the configurations $t = 0.15, s = 5$ and $t = 0.15, s = 15$ seem less promising than the convergence limit of the configuration $t = 0.25, s = 10$.

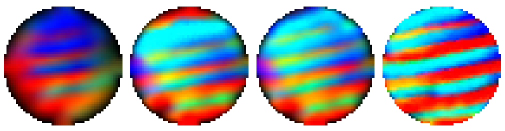
Figure 4.8b displays the resulting patches. The patches clearly present the pattern of blue and red stripes that we already encountered in previous experiments. Additionally we see that training on smaller filtering threshold values t increases the blur of a patch. Additionally we notice increasingly dark regions at the border of the patches for decreasing t.

Figure 4.8 displays the evaluation results for our patches when tested on the KITTI15 Training dataset. Overall the AEEs seem very promising. The patches trained with filtering thresholds $t \geq 0.15$ even reach AEEs that are comparable with tests against the undefended network in Section 4.3.1. Specifically, their AEEs to the grond truth are all larger than 40 pixels and they introduce an AEE to the unattacked prediction of more than 35 pixels. The large AEE to the ground truth estimation is an indicator that the large AEE to the ground truth flow is considerably influenced by the patch and not solely a result of excessive pre-processing activity.

The influence of the patches on the prediction result is visualized in Figure 4.9. We clearly see that all trained patches have a significant impact on the prediction result in their environment. Furthermore we notice that the patches remain largely unaffected even tough the ILP preprocessing has strong impact on the image. The patches thus seem to evade the detection mechanism for adversarial patches that is employed by ILP.



(a) Moving average of the training losses of our patch adversaries against FlowNetC with ILP pre-processing. The moving average was computed using a kernel of size 100.



(b) Perturbation optimization results for our ILP adversary introduced in Section 3.3.3 when trained against FlowNetC with ILP pre-processing using the training Algorithm 4.1. We set the blocksize across all experiments to $k = 16$ and the overlap to $o = 8$ as recommende by the authors [3].We set the inpainting radius to 5. Furthermore, and enumerated from left to right, we selected the additional defense parameters $(t = 0.08, s = 30), (t = 0.15, s = 5), (t = 0.15, s = 15), (t = 0.25, s = 5)$.

	$t = 0.08,$ $s = 30$	$t = 0.15,$ $s = 5$	$t = 0.15,$ $s = 15$	$t = 0.25$ $s = 10$
$AEE(F_{ILP}, \hat{F}_{ILP})$	12.541	36.583	39.119	35.613
$AEE(\hat{F}_{ILP}, F_{gt})$	25.141	45.100	47.402	44.374

Figure 4.8: Evaluation results of our adversary applied to FlowNetC with ILP pre-processing.

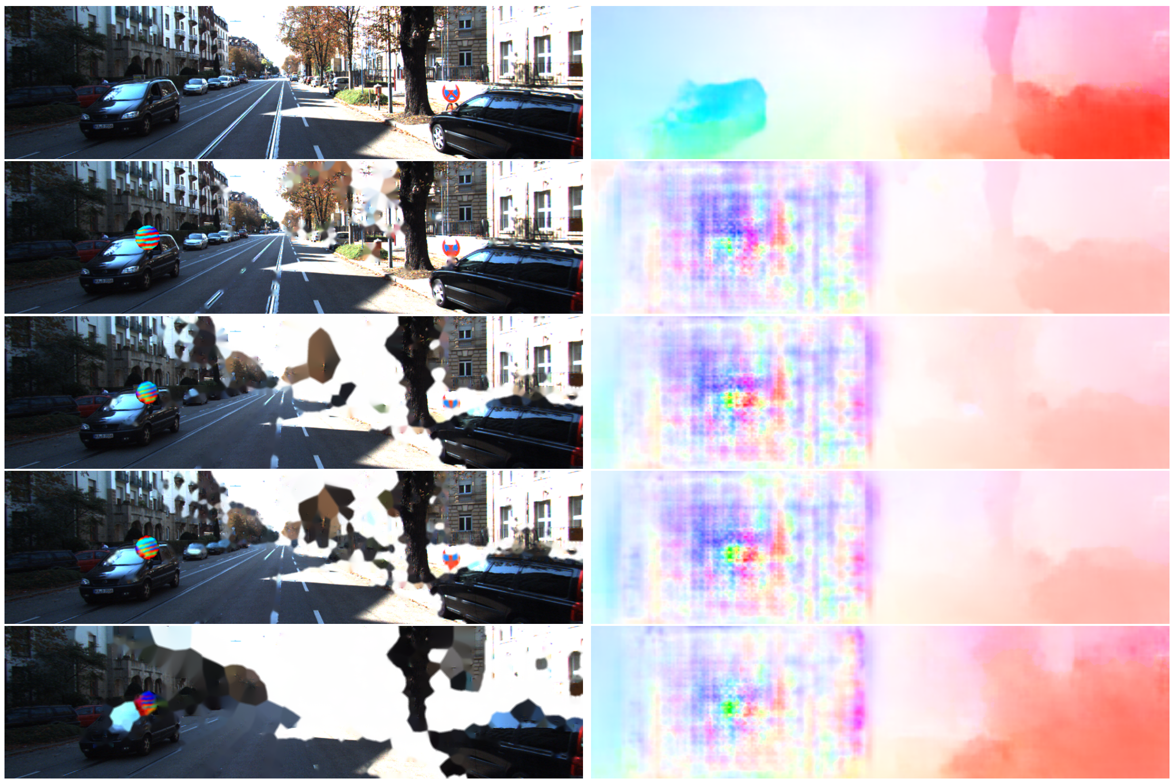


Figure 4.9: Comparison of the effect of the individual patches trained in experiment 3 on the flow field prediction. The first row shows the prediction of FlowNetC without any ILP pre-processing and attacks applied. The remaining rows show one attacked and then ILP pre-processed input image together with the resulting flow prediction.

4.4 Discussion

Our first experimental row against the undefended FlowNetC clearly reproduced previous results by Ranjan *et al.* [63]. Our results thus stress that adversarial patches of limited size are definitely able to confuse FlowNetC severely. Furthermore our first experimental row indicates that the patches trained by the official flowattack repository by Ranjan *et al.* [61] are either suboptimal or else we did not adjust the parameters correctly. As a result our patches performed consistently better than the ones trained with the Flowattack repository.

Additionally we examined the performance of various parameter ranges of t and s for ILP and LGS. For this we evaluated our best performing adversarial patch as well as the best performing patch trained with the Flowattack repository on FlowNetC with various parameter configurations. Our results are summarized in Figure 4.5b and Figure 4.5d. However in our experiments the parameter configuration $k = 15, o = 5, t = 0.1, s = 2.3$, recommended for LGS by Naseer *et al.* [52] seems suboptimal. On the other hand, the reportedly best performing parameters of Naseer *et al.* [52] were derived from image classification networks. Consequently our observation could indicate that the defense parameters are not transferable between different domains. One potential reason for this could be the differences in the network architecture.

The recommended parameters for ILP by Anand *et al.* [3] are however determined for optical flow networks. Nonetheless our results in Figure 4.5b indicate that their filtering threshold t might be too restrictive to filter unadaptively trained patches using our adversarial models. On the other hand, the both parameters represent sensible choices when defending against the patch trained using the Flowattack repository. One potential reason for this discrepancy is that Anand *et al.* [3] have optimized their parameters to patches that are generated using the Flowattack repository [61] specifically, because there seem to be only few alternatives.

The deflections that our attacks introduce into the predictions of the defense's with recommended parameters are significant. They introduce an AEE between the attacked and unattacked flow estimations of more than 25 pixels in our experiments. Furthermore our attacks generalize to more aggressive defense instances, that introduce noticeable bias into the images, to a large degree. As a consequence we consider the defenses considerably vulnerable against our adversarial patch attacks.

Generally, we observed in our experiments that the optimization procedure can have an influence on the optimization result. Specifically we noticed that our adversary which uses I-FGSM [39] updates consistently optimized towards a suboptimal patch when trained against FlowNetC with the LGS instance $t = 0.1, s = 2.3$. However because this parameter configuration performed best in the initial defense evaluation of the authors of LGS [52], we decided to change the adversary's update method and learning rate.

5 Outlook and Conclusion

The following two sections conclude this work. The first section outlines potentially promising research directions that are associated with or can build up on this work. The second section finally summarizes the contributions of this work and stresses key insights.

5.1 Outlook

Because we did not optimize our attacks for efficiency one could further examine the influence of more advanced optimization procedures on the efficiency of our generated patches. One could for instance apply better optimization update strategies with momentum or gradient descent [65] instead of the I-FGSM approach [39] that we used. Furthermore one could consider the reparameterization of the perturbation parameters with the approach that was originally published by Carlini and Wagner [16] and applied to perturbation attacks on optical flow networks by Schmalfuß *et al.* [67]. Also one could consider improved loss functions.

There are several respects in which one could extend on this work. Firstly, one can examine the transferability of our results to other optical flow networks that are vulnerable to the presented type of attacks. Considering the original work by Ranjan *et al.* [63] FlowNet2 [30] would be another interesting target. It however relies on a related architecture which might reduce the generality of additional insights. Additionally it would be interesting to apply related attacks on modern optical flow networks which are representing the state-of-the-art like RAFT [78]. This way one could further examine the robustness of advanced networks with respect to local input perturbations. In this context, one could also examine the influence of multiple static patches in a frame instead of one.

Additionally one could consider the transferability of defense concepts from image classification networks against input perturbation attacks to optical flow networks. This might be interesting because there seem to be no promising alternative pre-processing defenses established for optical flow networks. One possible approach could be to examine the influence of adversarial training on the network's robustness against global and local perturbation attacks. Additionally, one could optimize the ILP and LGS defenses to use information from image pairs jointly instead of separately.

Furthermore one could train a neural network to generate an efficient patch and its location for a given image pair. Such an adversary can be implemented as a convolutional neural network that receives an image sequence pair as an input and outputs suitably constrained perturbed images. Moreover one could also try to train a neural network to perform the estimation of manipulated image regions or to learn a suitable pre-processing operation.

5.2 Conclusion

In this work we examined the robustness of FlowNetC when it is equipped with the local pre-processing defenses ILP or LGS. We devised adversarial patch perturbation attacks based on the initial work of Ranjan *et al.* [63]. By consideration of the guidelines about adaptive attacks of Carlini *et al.* [14, 16] our attacks optimized against the defended optical flow networks. Additionally we applied concepts published by Athalye *et al.* [4] to restore obfuscated gradients.

We developed successful adaptive adversarial patch attacks against FlowNetC with ILP and LGS pre-processing. Our attacks introduce serious distortions for the recommended parameter configurations of both defenses. Furthermore we have demonstrated that our attacks generalize to stronger parameter configurations to some degree.

Consequently, our experiments indicate that the LGS and ILP pre-processing defenses are not sufficient to protect vulnerable optical flow networks against adversarial patch attacks. Thus we generally discourage the use of FlowNetC with any of both pre-processing defenses in security critical applications. Because of the absence of alternative defenses for vulnerable networks, one should instead employ optical flow networks which are robust against adversarial patch attacks as a result of their architecture as discussed by Schrodi *et al.* [68].

A Appendix

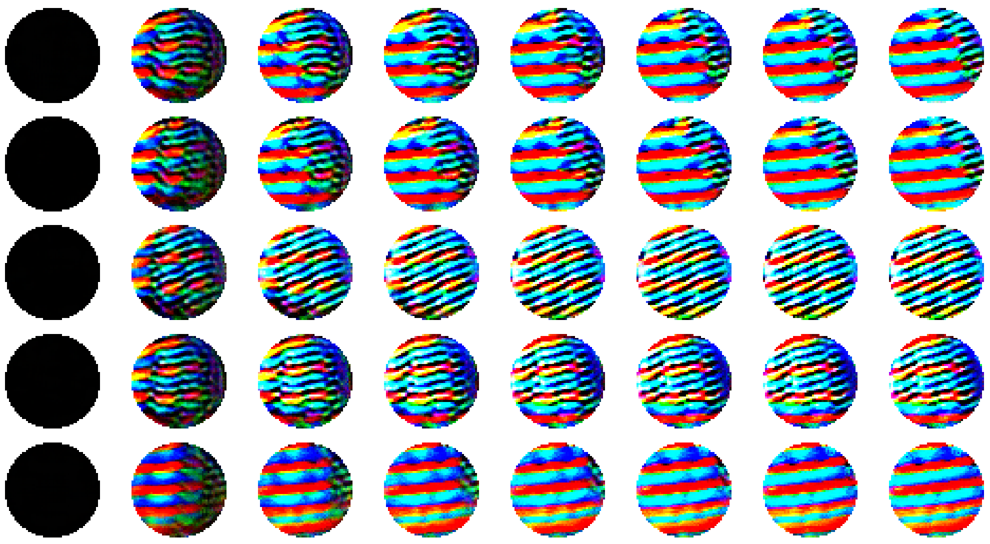


Figure A.1: Evolution of adversarial patches during training with our custom training procedure described in Algorithm 4.1. Every row represents a training run and every column represents a snapshot over time. The first row represents run 1. The first column shows the initialized patches in iteration 0. Subsequent columns show the patches after additional 1000 iterations.

Algorithm A.1 Revised version of the Fast Marching Method with included Inpainting Procedure by Telea [79]. We introduced several modifications to the algorithm in the original. Firstly we overloaded the function to compute the level-set distances for pixels in the close environment of the marked regions automatically before executing the original algorithm. Additionally we added explicit checks whether the indices are within the bounds of the image with $valid$. We assume KNOWN = 0, BAND = 1 and INSIDE = 2. HEAP is a heap datastructure that sorts its elements $(t, (a, b))$ for increasing t values. $Dilation(M, 3)$ performs a morphological dilation on M with a 3×3 stencil. Furthermore we assume $M \in \{0, 1\}^{H \times W}$ and $I \in [0, 1]^{H \times W}$. Telea mentioned that the algorithm can be generalized to color images by performing the procedure on each channel separately [79]. The formulation of this pseudocode is influenced by several implementations of the inpainting method by Telea [9, 26, 54, 79].

```
function FMM($I, M, d, T_{max}$)
    if $I! = None$ then
        $T_{out}$ := FMM($None, 1 - Dilation(M, 3), d, d$)
    end if
    $L := M + Dilation(M, 3)$
    $T := T_{out}$ if $I$ is not None else $abs(L - 1) \cdot INF$
    Initialize HEAP with pixels for which L == BAND
    while HEAP not empty do
        $t, (i, j)$ = head(HEAP)
        $L[i, j]$ = KNOWN
        for $(k, l) \in \{(i - 1, j), (i, j - 1), (i + 1, j), (i, j + 1)\}$ do
            if valid($k, l$) and $L[k, l]$ != KNOWN then
                if $L[k, l]$ == INSIDE then
                    $L[k, l]$ = BAND
                    $I[k, l]$ = INPAINT($k, l$) if $I$ is not None
                end if
                $t_{new} = \min($SOLVE($k - 1, l, k, l - 1$), SOLVE($k + 1, l, k, l - 1$),
                               SOLVE($k - 1, l, k, l + 1$), SOLVE($k + 1, l, k, l + 1$))
                if $T[k, l] > t_{new}$ and $t_{new} < T_{max}$ then
                    push(HEAP,($t_{new}, (k, l)$))
                    $T[k, l] = t_{new}$
                end if
            end if
        end for
    end while
end function
```

Algorithm A.2 Revised version of the Pixel Inpainting Procedure by Telea [79]. The function inpaints the pixel at the location i, j in the image I with an neighbourhood radius d, a level-set distances map T and labels L. Additionally the assumptions from the description of Algorithm A.1 apply. The formulation of this version of the inpainting procedure is based on the pseudo-code provided by Telea [79] and influenced by other published implementations [9, 26, 54].

```
function INPAINT($(i,j), I, T, L, d$)
    $Ia = 0$
    $s = 0$
    for $(k,l) \in \mathcal{N}_d(i,j)$ if $L[k,l] \neq$ INSIDE do
        $dT$ = 0 if not $valid(k+1,l)$ or not $valid(k-1,l)$ or
                    not $valid(k,l+1)$ or not $valid(k,l-1)$ else $\nabla T[k,l]$
        $dI$ = 0 if not $valid(k+1,l)$ or not $valid(k-1,l)$ or
                    not $valid(k,l+1)$ or not $valid(k,l-1)$ or
                    $L[k+1,l]$ == INSIDE or $L[k-1,l]$ == INSIDE
                    $L[k,l+1]$ == INSIDE or $L[k,l-1]$ == INSIDE else $\nabla I[k,l]$
        $r = (k-i, l-j)^\top$
        $dir = \frac{r}{\|r\|} \cdot dT$
        $dst = \frac{1}{\|r\|^2}$
        $lev = \frac{1}{1+\|T(k,l)-T(i,j)\|}$
        $w = dir \cdot dst \cdot lev$
        $Ia \mathrel{+}= w \cdot (I(k,l) + \nabla I \cdot r)$
        $s \mathrel{+}= w$
    end for
    return $\frac{Ia}{s}$
end function
```

Algorithm A.3 Revised version of the Solve Eikonal Equations Procedure by Telea [79]. The assumptions mentioned in the descriptions of Algorithm A.1 and Algorithm A.2 also apply here. The algorithm formulation presented here is based on the pseudo-code by Telea [79] and influenced by other published implementations of the algorithm [9, 26, 54].

```
function SOLVE($L, T, (i_1, j_1), (i_2, j_2)$)
    return $10^6$ if not $valid(i_1, j_1)$ or not $valid(i_2, j_2)$
    if $L[i_1, j_1]$ = KNOWN then
        if $L[i_2, j_2]$ = KNOWN then
            $r = \sqrt{2 - (T[i_1, j_1] - T[i_2, j_2])^2}$
            $s = \frac{T[i_1, j_1] + T[i_2, j_2] - r}{2}$
            if $s \geq T[i_1, j_1]$ and $s \geq T[i_2, j_2]$ then
                return $s$
            else
                $s \mathrel{+}= r$
                if $s \geq T[i_1, j_1]$ and $s \geq T[i_2, j_2]$ then
                    return $s$
                end if
            end if
        else
            return $1 + T[i_1, j_1]$
        end if
    else if $L[i_2, j_2]$ == KNOWN then
        return $1 + T[i_2, j_2]$
    end if
    return $10^6$
end function
```

Bibliography

[1] Robust vision challenge. http://www.robustvision.net/.

[2] Raft. `https://github.com/princeton-vl/RAFT`, 2021.

[3] Adithya Prem Anand, H Gokul, Harish Srinivasan, Pranav Vijay, and Vineeth Vijayaraghavan. Adversarial patch defense for optical flow networks in video action recognition. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1289–1296, 2020.

[4] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, July 2018.

[5] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293. PMLR, 2018.

[6] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International journal of computer vision*, 92(1):1–31, 2011.

[7] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

[8] Michael J Black and Padmanabhan Anandan. A framework for the robust estimation of optical flow. In *1993 (4th) International Conference on Computer Vision*, pages 231–236. IEEE, 1993.

[9] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[10] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.

[11] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *European conference on computer vision*, pages 25–36. Springer, 2004.

[12] Andrés Bruhn and Joachim Weickert. Lecture notes in computer vision, 2020.

[13] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, October 2012.

[14] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019.

[15] Nicholas Carlini and David Wagner. Defensive distillation is not robust to adversarial examples. *arXiv preprint arXiv:1607.04311*, 2016.

[16] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[17] Qifeng Chen and Vladlen Koltun. Full flow: Optical flow estimation by global optimization over regular grids. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4706–4714, 2016.

[18] Ping-yeh Chiang, Renkun Ni, Ahmed Abdelkader, Chen Zhu, Christoph Studer, and Tom Goldstein. Certified defenses for adversarial patches. In *8th International Conference on Learning Representations (ICLR 2020)(virtual)*. International Conference on Learning Representations, 2020.

[19] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3642–3649. IEEE, 2012.

[20] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[21] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. A study of the effect of jpg compression on adversarial images. *arXiv preprint arXiv:1608.00853*, 2016.

[22] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physical-world attacks on machine learning models. *arXiv preprint arXiv:1707.08945*, 2(3):4, 2017.

[23] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[24] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[25] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

[26] Eric Haines. jgt-code. `https://github.com/erich666/jgt-code/tree/master/Volume_09/Number_1/Telea2004`, 2020.

[27] Jamie Hayes. On visible adversarial perturbations & digital watermarking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1597–1604, 2018.

[28] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[29] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári. Learning with a strong adversary. *arXiv preprint arXiv:1511.03034*, 2015.

[30] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[31] Plotly Technologies Inc. Collaborative data science, 2015.

[32] Nathan Inkawhich, Matthew Inkawhich, Yiran Chen, and Hai Li. Adversarial attacks for optical flow-based action recognition classifiers. *arXiv preprint arXiv:1811.11875*, 2018.

[33] Danny Karmon, Daniel Zoran, and Yoav Goldberg. Lavan: Localized and visible adversarial noise. In *International Conference on Machine Learning*, pages 2507–2515. PMLR, 2018.

[34] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[35] Auguste Kerckhoffs. *La cryptographie militaire, ou, Des chiffres usités en temps de guerre: avec un nouveau procédé de déchiffrement applicable aux systèmes à double clef*. Librairie militaire de L. Baudoin, 1883.

[36] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[37] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[38] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *2011 International conference on computer vision*, pages 2556–2563. IEEE, 2011.

[39] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[40] Alexander Levine and Soheil Feizi. (de) randomized smoothing for certifiable defense against patch attacks. *Advances in Neural Information Processing Systems*, 33:6465–6475, 2020.

[41] Yingzhen Li, John Bradshaw, and Yash Sharma. Are generative classifiers more robust to adversarial attacks? In *International Conference on Machine Learning*, pages 3804–3814. PMLR, 2019.

[42] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[43] Jiajun Lu, Hussein Sibai, Evan Fabry, and David Forsyth. No need to worry about adversarial examples in object detection in autonomous vehicles. *arXiv preprint arXiv:1707.03501*, 2017.

[44] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[45] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. arXiv:1512.02134.

[46] K McLaren. Xiii—the development of the cie 1976 (l* a* b*) uniform colour space and colour-difference formula. *Journal of the Society of Dyers and Colourists*, 92(9):338–341, 1976.

[47] Lukas Mehl. flow_library. `https://github.com/cv-stuttgart/flow_library`, 2021.

[48] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[49] Moritz Menze, Christian Heipke, and Andreas Geiger. Discrete optimization for optical flow. In *German Conference on Pattern Recognition*, pages 16–28. Springer, 2015.

[50] metallurk. local_gradients_smoothing. `https://github.com/metallurk/local_gradients_smoothing`, 2020.

[51] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

[52] Muzammal Naseer, Salman Khan, and Fatih Porikli. Local gradients smoothing: Defense against localized adversarial attacks. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1300–1307. IEEE, 2019.

[53] Arild Nøkland. Improving back-propagation by adding an adversarial gradient. *arXiv preprint arXiv:1510.04189*, 2015.

[54] Olivier. pyheal. https://github.com/olvb/pyheal, 2020.

[55] Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. Improving adversarial robustness via promoting ensemble diversity. In *International Conference on Machine Learning*, pages 4970–4979. PMLR, 2019.

[56] Nicolas Papernot and Patrick McDaniel. On the effectiveness of defensive distillation. *arXiv preprint arXiv:1607.05113*, 2016.

[57] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[58] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.

[59] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[60] Clement Pinard. Pytorch correlation module. https://github.com/ClementPinard/Pytorch-Correlation-extension, 2020.

[61] Anurag Ranjan. flowattack. https://github.com/anuragranj/flowattack, 2020.

[62] Anurag Ranjan and Michael Black. Optical flow estimation using a spatial pyramid network. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017*, pages 2720–2729, Piscataway, NJ, USA, July 2017. IEEE.

[63] Anurag Ranjan, Joel Janai, Andreas Geiger, and Michael J. Black. Attacking optical flow. In *Proceedings International Conference on Computer Vision (ICCV)*, pages 2404–2413. IEEE, November 2019. ISSN: 2380-7504.

[64] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[65] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017.

[66] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[67] Jenny Schmalfuss, Philipp Scholze, and Andrés Bruhn. A perturbation constrained adversarial attack for evaluating the robustness of optical flow. *arXiv preprint arXiv:2203.13214*, 2022.

[68] Simon Schrodi, Tonmoy Saikia, and Thomas Brox. What causes optical flow networks to be vulnerable to physical adversarial attacks. *arXiv preprint arXiv:2103.16255*, 2021.

[69] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 32–36. IEEE, 2004.

[70] Ayon Sen, Xiaojin Zhu, Liam Marshall, and Robert Nowak. Should adversarial attacks use pixel p-norm? *arXiv preprint arXiv:1906.02439*, 2019.

[71] Sanchari Sen, Balaraman Ravindran, and Anand Raghunathan. Empir: Ensembles of mixed precision deep networks for increased robustness against adversarial attacks. *arXiv preprint arXiv:2004.10162*, 2020.

[72] James A Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996.

[73] 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*, pages 1528–1540, 2016.

[74] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[75] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. 2018.

[76] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[77] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[78] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020.

[79] Alexandru Telea. An image inpainting technique based on the fast marching method. *Journal of graphics tools*, 9(1):23–34, 2004.

[80] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1633–1645. Curran Associates, Inc., 2020.

[81] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[82] Joost van de Weijer and Th Gevers. Robust optical flow from photometric invariants. In *2004 International Conference on Image Processing, 2004. ICIP'04.*, volume 3, pages 1835–1838. IEEE, 2004.

[83] Gunjan Verma and Ananthram Swami. Error correcting output codes improve probability estimation and adversarial robustness of deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[84] Alex Wong, Mukund Mundhra, and Stefano Soatto. Stereopagnosia: Fooling stereo networks with adversarial perturbations. *arXiv preprint arXiv:2009.10142*, 2020.

[85] Benjamin Wortman. Hidden patch attacks for optical flow. In *ICML 2021 Workshop on Adversarial Machine Learning*, 2021.

[86] Chong Xiang, Arjun Nitin Bhagoji, Vikash Sehwag, and Prateek Mittal. {PatchGuard}: A provably robust defense against adversarial patches via small receptive fields and masking. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2237–2254, 2021.

[87] Chong Xiang and Prateek Mittal. Patchguard++: Efficient provable attack detection against adversarial patches. *arXiv preprint arXiv:2104.12609*, 2021.

[88] Chang Xiao, Peilin Zhong, and Changxi Zheng. Enhancing adversarial defense by k-winners-take-all. *arXiv preprint arXiv:1905.10510*, 2019.

[89] Tianwei Zhang, Huayan Zhang, Yang Li, Yoshihiko Nakamura, and Lei Zhang. Flowfusion: Dynamic dense rgb-d slam based on optical flow. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7322–7328. IEEE, 2020.

[90] Ziqi Zhang, Xinge Zhu, Yingwei Li, Xiangqun Chen, and Yao Guo. Adversarial attacks on monocular depth estimation. *arXiv preprint arXiv:2003.10315*, 2020.

All links were last followed on April 10, 2022.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

place, date, signature