





Interactive Control over Temporal Consistency while Stylizing Video Streams

Sumit Shekhar^{1*}, Max Reimann^{1*}, Moritz Hilscher¹, Amir Semmo^{1,2},
Jürgen Döllner¹, and Matthias Trapp¹

¹Hasso Plattner Institute for Digital Engineering, University of Potsdam, Germany

²Digital Masterpieces GmbH, Germany

(* denotes equal contribution)

Abstract

Image stylization has seen significant advancement and widespread interest over the years, leading to the development of a multitude of techniques. Extending these stylization techniques, such as Neural Style Transfer (NST), to videos is often achieved by applying them on a per-frame basis. However, per-frame stylization usually lacks temporal consistency, expressed by undesirable flickering artifacts. Most of the existing approaches for enforcing temporal consistency suffer from one or more of the following drawbacks: They (1) are only suitable for a limited range of techniques, (2) do not support online processing as they require the complete video as input, (3) cannot provide consistency for the task of stylization, or (4) do not provide interactive consistency control. Domain-agnostic techniques for temporal consistency aim to eradicate flickering completely but typically disregard aesthetic aspects. For stylization tasks, however, consistency control is an essential requirement as a certain amount of flickering adds to the artistic look and feel. Moreover, making this control interactive is paramount from a usability perspective. To achieve the above requirements, we propose an approach that stylizes video streams in real-time at full HD resolutions while providing interactive consistency control. We develop a lite optical-flow network that operates at 80 Frames per second (FPS) on desktop systems with sufficient accuracy. Further, we employ an adaptive combination of local and global consistency features and enable interactive selection between them. Objective and subjective evaluations demonstrate that our method is superior to state-of-the-art video consistency approaches. maxreimann.github.io/stream-consistency

CCS Concepts

• **Computing methodologies**, . . . , **Image-based rendering**; **Non-photorealistic rendering**; **Image processing**;

1. Introduction

For thousands of years, paintings have served as a tool for visual communication and expression. However, it was not until the late 20th century that computers were used to simulate paintings [Hae90]. In the course of following decades, the field of artistic stylization [KCWI13] has significantly developed and extended by learning-based methods, such as Neural Style Transfers (NSTs) [SID17, JYF*20]. Even though a large number of image stylization techniques exist, extending these to video remains challenging. A major obstacle in this regard is the enforcement of temporal coherence between stylized video frames. With the proliferation of video streaming applications, stylizing video streams has also become popular, however, the requirements of low-latency processing add additional challenges. Most of the existing methods, to address the above, can be classified into one of the following four categories:

Style Specific. A common approach is to develop a specific method for a particular artistic style and exploit its characteristics for temporal coherency [BNTS07, NSC*11]. Such meth-

ods work effectively for the specific target style, however, do not generalize well. Many of these specialized approaches have been discussed by Bénard *et al.* [BTC13].

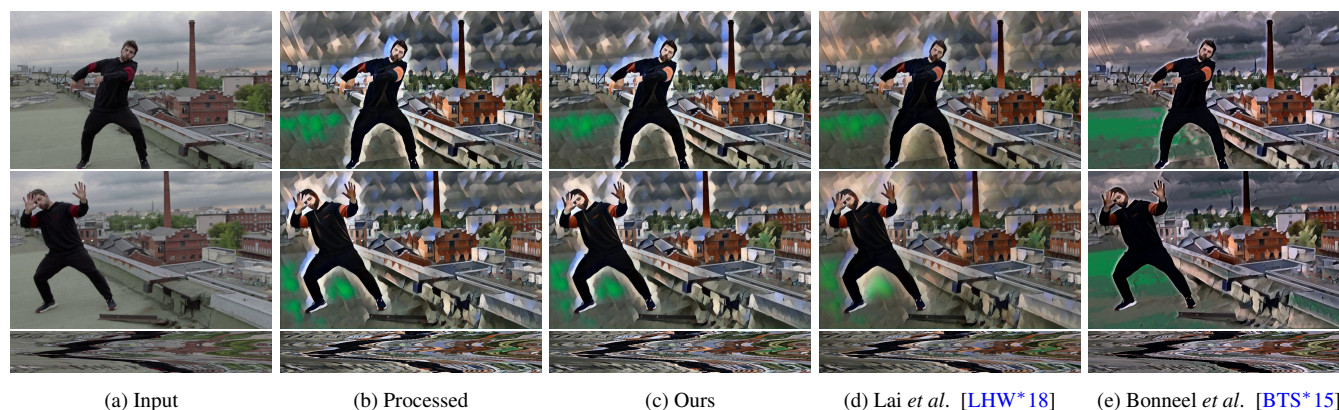
Coherent Noise. Another class of techniques adopts and transforms a generic, temporally-coherent noise function to yield a visually plausible stylized output [BLV*10, KP11]. Compared to target-based coherence enforcement [BNTS07], these apply to a wider range of techniques but are limited to scenarios with rapid temporal changes.

Stylization by Example. More recently, authors have adopted a stylization-by-example approach to support a wide range of stylization techniques [BCK*13, JST*19, TFK*20, FKL*21]. However, this approach requires the paring of the complete video and keyframe marking. Thus, by design, it does not apply to video streams.

Consistent Video Filtering. One can also enable the stylization of video streams using consistent video filtering techniques. Existing approaches are either not well-suited for Image-based Artistic Rendering (IB-AR) [BTS*15, YCC17] (Fig. 1) or do not provide interactive consistency control [LHW*18, TDKP21],

Table 1: Comparing existing consistent video filtering methods with ours with regards to consistency control. Here, the color green denotes the aspect which is favorable to interactive consistency-control while the color red denotes otherwise (“N/A” denotes Not-Applicable).

Aspects	Bonneel et al. [BTS*15]	Yao et al. [YCC17]	Lai et al. [LHW*18]	Shekhar et al. [SST*19]	Thiemonier et al. [TDKP21]	Ours
Requires pre-processing?	No	Yes	No	Yes	No	No
Provides consistency control at inference time?	Yes	No	No	Yes	No	Yes
Provides interactive consistency control?	No	N/A	N/A	Yes	N/A	Yes

**Figure 1:** For the top-row: first two columns depict (a) input and (b) processed result for frame-24, columns three to five depict the corresponding consistent output using (c) Ours (d) Lai’s, and (e) Bonneel’s method. For the mid-row: depict the corresponding results for frame-80. For the bottom row: we show the Temporal Slice Image (TSI) for the entire video sequence depicting long-term temporal similarity with the per-frame processed output. Note, that our method is able to preserve the look and feel of the per-frame processed result in comparison to the method of Lai et al. which suffers from color bleeding artifacts while the stylized textures are lost for the output of Bonneel et al. . Please see the supplementary material for video results.

which is an essential requirement for artistic rendering [FLJ*14]. Currently, the only method that provides interactive consistency control is limited to offline processing and requires pre-processing [SST*19].

We aim to develop a temporal consistency enforcement approach for artistic stylization techniques that provides (1) interactive consistency control and (2) online processing to facilitate the application to video streams.

A determining factor towards the slow performance of existing online and interactive consistent video filtering technique [BTS*15] is the costly step of the optical-flow computation. Previous works using learning-based methods are able to achieve a considerable accuracy for optical-flow estimation [TD20, JCL*21]. However, we argue that such high accuracy is not particularly necessary to enforce temporal consistency for artistic stylization tasks. To validate our conjecture, we conduct a user study, wherein the participants prefer the final consistent video output generated using our flow network as compared to that being obtained using State-of-the-art (SOTA) approaches.

In contrast to accuracy, less attention has been paid to improving the run-time performance of optical-flow estimation, which is essential for online-interactive editing. To this end, we develop a lite optical-flow neural network that runs at a high-speed (approx. 80 FPS on mid-tier desktop GPUs) while maintaining sufficient accuracy. The compact network is also deployable on mobile devices (iPhones and iPads) where it runs at interactive frame rates (24 FPS on iPad Pro 2020). We use the optical-flow output from

the above network to warp neighboring processed frames (for local consistency) and previous consistent output (for global consistency), which allows for interactive global and local temporal consistency control. Our approach is able to stabilize incoming video streams in real-time with one frame latency on a consumer desktop GPU at HD resolutions, and, using a fast preset, also in full HD.

To summarize we present the following contributions:

1. A novel approach for making per-frame stylized videos temporally consistent via an adaptive combination of local and global consistency features which allows for interactive consistency control.
2. A lite optical-flow network, to achieve interactive performance, that runs at 80 FPS on a mid-tier desktop PC and at 24 FPS on a mobile device while achieving reasonable accuracy.

Note, that we define artistic stylization as the adaptation of colors, textures, and strokes. While our approach is effective for most image-based stylization techniques (e.g., NSTs, algorithmic filtering), it cannot handle significant shape or content inconsistencies between frames introduced by semantically-driven image synthesis (e.g., image-to-image diffusion-based models [RBL*22]). Flow-based warping is insufficient to enforce consistency in such cases.

2. Background & Related Work

Consistent Video Filtering. Lang et al. [LWA*12] propose a solution to enforce temporal consistency for a large class of

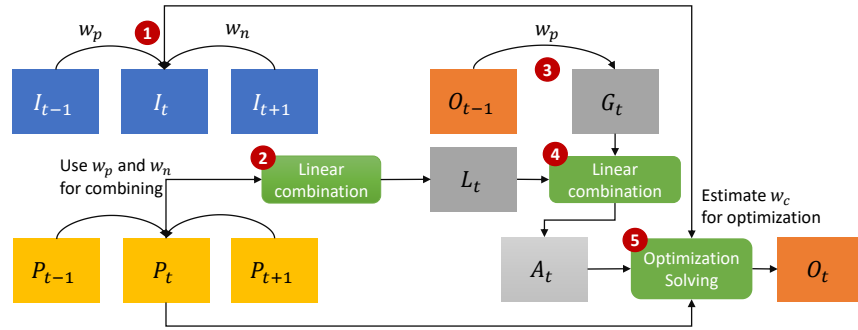


Figure 2: Schematic overview of our approach: (1) We start by calculating the warping weights w_p and w_n by applying Eqn. 3 on the input image sequence I_{t-1}, I_t, I_{t+1} . (2) The computed weights are used to linearly combine the per-frame processed sequence P_{t-1}, P_t, P_{t+1} to obtain the locally consistent image L_t , see Eqn. 2. (3) To obtain the globally consistent version G_t we warp the output at previous time instance O_{t-1} as depicted in Eqn. 4. (4) The local and global consistent images, L_t and G_t , are linearly combined to obtain a temporally smooth version A_t , see Eqn. 5. (5) To include high-frequency details from the per-frame processed result, A_t and P_t are adaptively combined via the optimization in Eqn. 1 using the weights w_c (Eqn. 7) to obtain the final result O_t .

optimization-based problems via iterative filtering along the motion path. Dong *et al.* [DBZY15] address the problem of temporal inconsistency for enhancement algorithms by dividing individual video frames into multiple regions and performing a region-based spatio-temporal optimization. Bonneel *et al.* [BTS*15] was the first to present a generalized approach for consistent video filtering which is agnostic to the type of filtering applied on individual video frames. The method combines gradient-based characteristics of the per-frame processed result with the warped version of the previous-frame output using a gradient-domain-based optimization scheme. Yao *et al.* [YCC17] propose a similar approach however considers multiple key-frames for warping-based consistency to avoid problems due to occlusion. Both of the approaches assume that the gradient of the processed video is similar to that of the input video and thus cannot handle artistic rendering tasks where new gradients resembling brush strokes are generated as part of the stylization process. Moreover, due to slow optical-flow computation, they are non-interactive in nature. Shekhar *et al.* [SST*19] employs a similar formulation as Bonneel *et al.*, with the difference of using a temporally denoised version of the current frame for consistency guidance. However, the temporal denoising requires the complete video as input making the method offline in nature. Lai *et al.* [LHW*18] propose the first learning-based technique in this context. The authors employ perceptual loss to enforce similarity with the processed frames and for consistency make use of short-term and long-term temporal losses. Thimonier *et al.* [TDKP21] employs a ping-pong loss and a corresponding training procedure for temporal consistency. Both learning-based techniques are faster than their optimization-based counterpart since they do not perform optical-flow computation at inference time. However, these learning-based techniques do not allow to control of the degree of consistency in the final output which is vital for the task of stylization. Thus, the above-discussed methods are either non-interactive/offline or do not provide any consistency control at inference time. Our approach addresses these limitations (Tab. 1).

Optical Flow for Consistent Filtering. Both Bonneel *et al.* and Yao *et al.* use the PatchMatch algorithm [BSFG09] for flow-based

warping, however, the slow performance of PatchMatch makes them non-interactive. Lai *et al.* use FlowNet 2.0 [IMS*17] for flow-based warping to design their short-term and long-term temporal consistency losses. FlowNet 2.0 is on par with the quality of state-of-the-art classical methods, however, due to a large number of parameters and operations, achieves only interactive frame rates even on high-end desktop Graphical Processing Units (GPUs). An improved compact optical-flow Convolutional Neural Network (CNN) is proposed by Sun *et al.* [SYLK18] – PWC-Net. It combines coarse-to-fine estimation with pyramidal image features, correlation, warping, and CNN-based estimation. Furthermore, a refinement CNN is stacked at the end to improve the final flow estimate. PWC-Net is orders of magnitude smaller than FlowNet 2.0 and runs at real-time frame rates using desktop GPUs. Liu *et al.* [LZH*20] employ their approach to train a similar architecture in an unsupervised setting and achieve reasonable accuracy – ARFlow. LiteFlowNet and its successor LiteFlowNet2, both proposed by Hui *et al.* [HTL18, HTL20], have similar compact architectures. Further improvement in accuracy is achieved by models using iterative refinements, such as RAFT [TD20] and transformer modules such as GMA [JCL*21], however, they heavily trade runtime for accuracy. Based on a runtime-accuracy comparison (see Sec. 3.2), we select PWC-Net as a base network to develop a "Lite" flow network with improved performance for interactive consistent filtering.

Temporal Consistency for Video Stylization. Litwinowicz [Lit97] describes a technique to apply an impressionist effect on images and videos. For enforcing temporal coherence, optical flow was used to transform the brush strokes from one frame to the next. Winnemöller *et al.* [WOG06] develop a real-time video and image abstraction framework. The authors employ soft quantization that spreads over a larger area, thus significantly reducing temporal incoherence. Bousseau *et al.* [BNTS07] advects texture in forward and backward directions using optical flow for coherent water-colorization of videos. Noris *et al.* [NSC*11] preserve the geometric richness of the sketched style in each frame while allowing to successful decrease the temporal noise to a desirable

rate. Fišer *et al.* [FLJ*14] propose similar temporal noise control for hand-colored animations. Noris *et al.* and Fišer *et al.* come close to our vision of providing control over the extent of temporal noise. However, the method of Noris *et al.* can only handle sketchy animations while Fišer *et al.* requires a clean animation as input onto which temporal noise, extracted from hand-colored examples, is added. In comparison, our approach works on already stylized videos and can handle a broad range of stylization techniques. Numerous such specialized video-based approaches have been discussed by Bénard *et al.* [BTC13]. The above classical IB-AR techniques approximate rendering primitives by modifying traditional image filters. Most often, they use low-level image features for modeling and fail to model structures resembling a particular style. Recently, deep CNNs were successfully used to transfer high-level style attributes from a painting onto a given image [GEB16]. Various methods have been proposed to extend the above for videos [HWL*17, CLY*17, GJAF17, RDB18, LLKY19, PP19, DTD*21]. Ruder *et al.* [RDB18] proposes a novel initialization technique and loss functions for consistent stylized output even in cases with large motion and strong occlusion. The methods of Gupta *et al.* [GJAF17], Chen *et al.* [CLY*17], and Huang *et al.* [HWL*17] enforce consistency via certain formulations of temporal loss and use optical-flow based warping only during the training phase thus achieving fast performance. Li *et al.* [LLKY19] proposes a method for arbitrary style transfer and shows its applicability to real-time video style transfer by applying style features to consecutive frames using a shallow autoencoder. However, we show that our approach, applied to their per-frame processed videos is able to significantly reduce flickering and is more consistent than their stabilized version (see supplementary). Puy and Pérez [PP19] develop a flexible deep CNN for controllable artistic style transfer that allows for the addition of a temporal regularizer at testing time to remove the flickering artifacts. The above method comes closest in terms of providing some consistency control at test time for NST-based methods. However, they cannot handle classical stylization techniques. Keyframe-based Stylization (KBS) [BCK*13, JST*19, TFK*20, FKL*21] caters to both classical and neural paradigms via priors involving keyframe-based warping. Nonetheless, it is usually applied as an offline process involving pre-training on the input video. Moreover, we show that our approach is able to interactively stabilize online KBS approaches such as [TFK*20]. We aim to propose a generic solution that is agnostic to the type of stylization and provides online performance and interactive consistency control.

3. Method

3.1. Temporal Consistency Enforcement

Given an input video stream $\dots I_{t-1}, I_t, I_{t+1}, \dots$ and its per-frame processed version $\dots P_{t-1}, P_t, P_{t+1}, \dots$, we seek to find a temporally consistent output $\dots O_{t-1}, O_t, O_{t+1}, \dots$. Our method is agnostic to the stylization technique f applied to each frame, where $P_t = f(I_t)$. However, it is necessary for f to not introduce significant shape or content inconsistencies between consecutive frames, as the changes in the stylized frames should correspond to the optical flow (calculated based on the content). We initialize the consistent output for the first frame as its per-frame processed result

Table 2: Constituent elements of smoothness term in Eqn. 1 for different methods. Here, w_s and T_d refers to saliency-based weights and temporally-denoised image respectively, introduced by Shekhar *et al.* [SST*19]

Method	Weight	Consistent Image
Ours	w_c	A_t
Bonneel <i>et al.</i> [BTS*15]	w_p	$\Gamma(O_{t-1})$
Shekhar <i>et al.</i> [SST*19]	w_s	T_d

i.e., $O_1 = P_1$. To obtain the output for subsequent frames (O_t at any given instance t) we require only a snippet of input (I_{t-1}, I_t, I_{t+1}) and processed streams (P_{t-1}, P_t, P_{t+1}), and the consistent output at the previous instance O_{t-1} . For enforcing consistency, we solve the following gradient-domain optimization scheme:

$$E(O_t) = \int_{\Omega} \left(\underbrace{\|\nabla O_t - \nabla P_t\|^2}_{\text{data}} + \underbrace{w_c \|O_t - A_t\|^2}_{\text{smoothness}} \right) d\Omega. \quad (1)$$

where Ω represents the image domain. The *data* term in this optimization enforces similarity with the per-frame processed result P_t in the gradient domain. The gradient-based data term ensures that we borrow only the necessary details from the per-frame processed results (in the form of edges) while avoiding inconsistencies. Thus, high-frequency details are taken from P_t and the *smoothness* term enforces temporal consistency where low-frequency content is taken from the image A_t . The optimization formulation in Eqn. 1 is commonly known as *screened Poisson equation* and has been successfully employed for various image editing applications [BCCZ08, BZCC10]. In the context of consistent video filtering, it was first used by Bonneel *et al.* [BTS*15] followed by Shekhar *et al.* [SST*19] (Tab. 2). However, our novelty is the way in which we construct our *smoothness* term that, unlike previous approaches, considers both *global* and *local* consistency aspects. Our novel smoothness term is able to better preserve the color and textures in the stylized output while providing both short-term and long-term temporal consistency.

Local Consistency. For enforcing temporal consistency at a local level, we use optical flow to warp neighboring per-frame processed results to the current time instance t . This is performed by computing an adaptive combination of (1) warped previous per-frame processed image $\Gamma(P_{t-1})$, (2) warped next per-frame processed image $\Gamma(P_{t+1})$, and (3) the current per-frame processed image P_t , where Γ is the warping function. By including both backward and forward warping in our formulation, we are able to significantly reduce artifacts due to occlusion and flow inaccuracies. The linear combination of (1), (2), and (3) gives us a locally consistent version L_t where,

$$L_t = (1 - (w_p + w_n)) \cdot P_t + w_p \cdot \Gamma(P_{t-1}) + w_n \cdot \Gamma(P_{t+1}). \quad (2)$$

The weights w_p and w_n capture the inaccuracies in the warping of previous and next frames respectively and are defined as follows:

$$\begin{aligned} w_p &= \exp\left(-\alpha \|I_t - \Gamma(I_{t-1})\|^2\right) \text{ and} \\ w_n &= \exp\left(-\alpha \|I_t - \Gamma(I_{t+1})\|^2\right). \end{aligned} \quad (3)$$

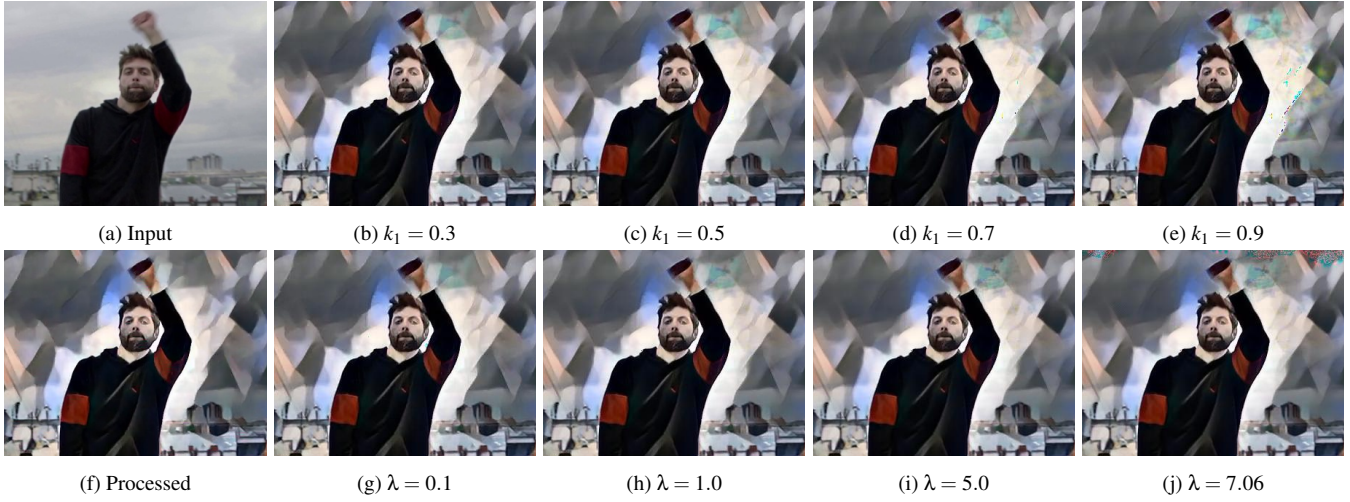


Figure 3: The level of consistency in the final output can be controlled via parameters k_1 and λ . Here we show how the final result varies by increasing these, for lower values the consistency is negligible and the results (Fig. 3b and Fig. 3g) visually look similar to the per-frame processed output (Fig. 3b). For higher values, we start observing artifacts due to ghosting and/or optimization (Fig. 3e and Fig. 3j).

In order to also incorporate contribution from P_t , we clamp the weights w_p and w_n as follows: $w_p \in [0, k_1]$ and $w_n \in [0, k_2]$, where k_1 and k_2 are two constants and their sum is less than one, i.e., $0 < (k_1 + k_2) < 1$. The locally consistent image sequence given by L_t has improved temporal consistency over the per-frame processed output, however, it still has visible flickering artifacts. Thus, the reduction in flickering due to the warping of only one temporal neighbor is not sufficient. To further improve consistency, one can warp more neighboring frames around the current time instance t . As we increase the temporal window size for such an adaptive combination it has a denoising effect leading to further reduction in flickering. The temporal denoising performed by Shekhar et al. [SST*19], for enforcing consistency, can be considered as a specific example of the above scenario. However, for interactive stylization, warping more frames to the current instance is not feasible due to time constraints. Moreover, in the case of video streams, we do not have frames to warp from the forward temporal direction.

Global Consistency. In order to overcome this limitation, existing techniques [BTS*15, LHW*18] adopt a global approach. For global consistency, one can consider the previous stabilized output O_{t-1} and enforce similarity with its warped version G_t where,

$$G_t = \Gamma(O_{t-1}). \quad (4)$$

To enforce only global temporal smoothness, we replace A_t with G_t in Eqn. 1. Further, in order to compensate for optical-flow inaccuracies, the smoothness term is weighted using w_p (i.e., $w_c = w_p$) in Eqn. 1.

However, considering only global consistency for flicker reduction leads to a loss of stylization (in terms of colors and textures) and local temporal variations in the final output. Moreover, in this case, any warping error (due to flow inaccuracies) or noise (as part of the stylization process) keeps getting propagated to future

frames. Due to the above factors, such an approach only gives plausible results where the gradients of the original video are similar to the gradients of the processed video. The above does not hold for the task of stylization where stylistic elements such as brush strokes, textures, or stroke textures [ZGWX05], in general, can vary largely between frames even for small changes in input gradient.

Combining Local and Global Consistency. For preserving local temporal variations (in terms of look and feel) while significantly reducing flickering artifacts, we linearly combine globally and locally consistent images G_t and L_t respectively,

$$A_t = w_p \cdot G_t + (1 - w_p) \cdot L_t. \quad (5)$$

We use the adaptively combined image A_t as our reference for consistency while enforcing temporal smoothness in Eqn. 1. The upper limit of weight w_p (i.e., k_1) can be increased to increase the influence of global-temporal smoothness and vice versa. Further, the influence of the smoothness term is controlled by per-pixel consistency weights w_c . We would like to invoke the smoothness term only when the warping accuracy is sufficiently high. To this end, we construct a warped version of the input image similar to L_t as,

$$A_t^I = (1 - (w_p + w_n)) \cdot I_t + w_p \cdot \Gamma(I_{t-1}) + w_n \cdot \Gamma(I_{t+1}). \quad (6)$$

Only when the input image I_t is similar to A_t^I , the smoothness term is invoked. To measure this similarity, we use the weight w_c ,

$$w_c = \lambda \cdot \exp\left(-\alpha \|I_t - A_t^I\|^2\right). \quad (7)$$

The parameter λ is used to scale up or down the weight w_c .

Consistency Control Modes. The above adaptive combination of local and global consistency provides two different ways of consistency control in the final output. By increasing the upper limit of w_p , i.e., k_1 we can increase the proportion of global consistency in the adaptively combined image A_t and vice versa. On the other

hand, the optimization parameter λ dictates how close the output O_t will be to the adaptively combined image A_t . Thus, the level of consistency in the final output can be controlled in two different ways: (1) by setting the upper limit of parameter w_p , i.e., k_1 or (2) by scaling the weight parameter λ . For low values of k_1 (Fig. 3b), the consistency enforced is negligible and the final result resembles the per-frame processed output (Fig. 3f). However, for higher values, we start observing noisy ghosting artifacts (Fig. 3e). The higher values for k_1 translate to using only global consistency which results in the accumulation of flow inaccuracies visualized as ghosting artifacts. Similarly, for lower values of λ (Fig. 3g), the final result is visually similar to the per-frame processed output (Fig. 3f). However, for higher values, the optimization becomes unstable resulting in noisy optimization-based artifacts. (Fig. 3j).

Optimization Solver. The energy terms in Eqn. 1 are smooth and convex in nature, which allows a straightforward energy minimization with respect to O_t . To this end, we employ an iterative approach thus avoiding: (i) storage of a large matrix in memory and (ii) further estimating its inverse. Moreover, an iterative approach allows us to stop the solver once we have achieved visually plausible results. An iterative update O_t^{j+1} is obtained by employing Stochastic Gradient Descent (SGD) with momentum [Qia99],

$$O_t^{j+1} = O_t^j - \eta \nabla E(O_t^j) + \kappa(O_t^j - O_t^{j-1}). \quad (8)$$

where η and κ are the step size parameters, ∇E is the energy gradient with respect to O_t , and j is the iteration count. For most of our experiments, $\eta = 0.15$ and $\kappa = 0.2$ yield plausible results. We consider the trade-off between performance vs. accuracy as stopping criteria and do not compute energy residue for this purpose. To obtain a consistent output while having interactive performance, we empirically determine 150 iterations to be sufficient. The optimization is stable for the given parameter settings and early stopping is only employed for computational gain.

An integral aspect common to both our *local* and *global* consistency is the warping function Γ . Apart from the number of solver iterations, for interactive performance the above warping should also happen at a fast rate – which in turn necessitates fast optical-flow estimation.

3.2. Lite Optical-Flow Network

We aim to obtain a flow network capable of running at high-speed on consumer hardware with reasonable accuracy. To this end, we start by selecting an existing CNN-based optical flow estimation technique, based on accuracy vs. run-time analysis. After the selection of a base network, we perform further optimization steps to increase the performance as outlined in Fig. 4.

Base Network Selection for Compression. In Fig. 5, we compare several well-known optical methods to find a base network candidate that best matches our runtime/accuracy requirements. We employ the following models for this: FlowNet 2.0 [IMS*17], SpyNet [RB17], LiteFlowNet2 [HTL20], PWCNet [SYLK18], ARFlow [LZH*20], VCN [YR19a], RAFT [TD20] and finally GMA [JCL*21] (state-of-the-art in terms of EPE-based accuracy). Our experiments are carried out on an Nvidia RTX 2070 GPU,

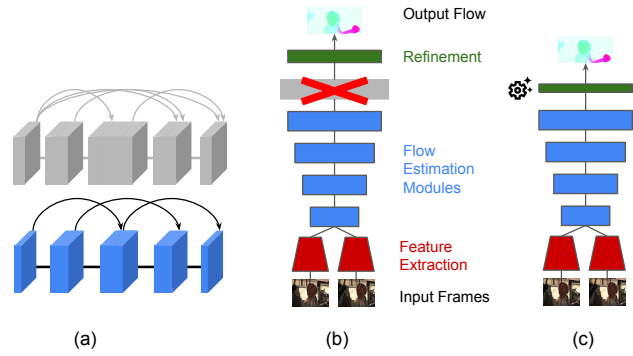


Figure 4: Modification of the PWC-Net [SYLK18] architecture for real-time performance. We apply the following network compression steps: (a) Replace DenseNet connections with light ones, (b) Reduce the number of flow estimators, and (c) Replace dense connections in the refinement module with separable convolutions.

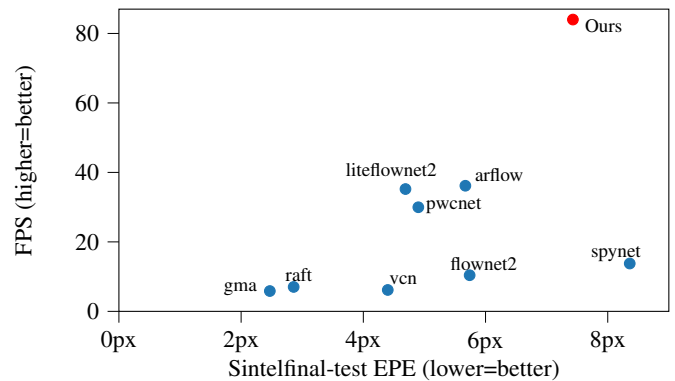
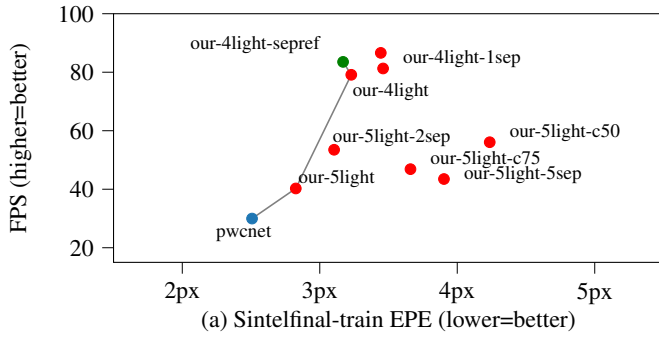


Figure 5: Accuracy vs. run-time performance of existing methods measured on Sintel Final (Test set) [BWSB12]. The Endpoint Error (EPE) metric measures Euclidean distance (in pixels) between ground truth and predicted optical flow vectors. Note how our method achieves a high FPS while being accurate enough for temporal consistency enforcement.

which we deem to be a good representative of a current mid-to higher-end consumer GPU. Under a constraint of interactive performance on consumer hardware, LiteFlowNet2 [HTL20] and PWC-Net [SYLK18] offer the best trade-off between run-time performance and accuracy (Fig. 5). LiteFlowNet2 [HTL20] is already an optimized version of FlowNet 2.0 [IMS*17], in comparison PWC-Net [SYLK18] has more potential for optimization/compression. Moreover, recently it has been shown that PWC-Net can achieve similar accuracy to RAFT when trained on a large-scale synthetic dataset [SVH*21] and that PWC-Net achieves favorable trade-offs vs. other state-of-the-art methods when selecting for runtime performance or higher image resolutions [SHR*22]. Hence, we select PWC-Net for further compression.



Modifier	Description	Default
-Nlight	N light [LZH*20] flow estimators.	5 dense [SYLK18]
-Msep	last M flow estimators use depthwise separable convolutions [HZC*17].	standard convs.
-sepref	refinement uses depthwise separable convolutions [HZC*17].	standard convs.
-cP	use $P\%$ of channels.	100%

(b) Legend of our CNN variants.

Figure 6: Accuracy vs. run-time performance of our CNN variants on desktop, measured on Sintel Final (Train) [BWSB12]. Optimization steps that lead to significant improvement in run-time are connected by a line. Our architectural modifications to PWC-Net [SYLK18] are detailed on the right, e.g., our-4light-sepref denotes a 4 light flow estimators and refinement using depthwise separable convolutions. We achieve a high accuracy on Sintel training data, however, for testing data the accuracy is low, see Fig. 5.

Optimized Network Architecture. We start with the base architecture of PWC-Net. As the first compression step, we reduce the computationally expensive DenseNet [HLvdMW17] connections in the flow estimators to retain connections only in the last two layers ("light" in Fig. 6b). Similar to LiteFlowNet2 [HTL20], we remove the fifth flow estimator – operating on the highest resolution – as it heavily trades off run-time for only a marginal increase in accuracy (compare "4light" vs "5light" in Fig. 6b). We replace the standard convolutions in the refinement by depthwise separable convolutions [HZC*17] ("-sepref" in Fig. 6b). Moreover, we also explore reducing the number of channels [HZC*17], but find that reducing channels results in a worse trade-off as compared to other optimizations.

Training. For training, we follow the original PWC-Net [SYLK18] schedule. However, we find that weighting the multi-scale losses equally, instead of exponentially [SYLK18, HTL18, HTL20, YR19a], improves accuracy. For our experiments on the desktop system, we use PyTorch [P*19] and take inspiration from the implementation by Niklaus [Nik18]. Similar to PWC-Net [SYLK18], we train our mobile architecture on the training dataset schedule FlyingChairs [FDI*15] → FlyingThings3D [MIH*16] → Sintel [BWSB12]. In the supplementary material, we provide training settings for each stage in detail. We employ a multi-scale loss [SYLK18] applied to each flow estimator and optimize using the AdamW optimizer [LH19] with $\beta_1 = 0.09$, $\beta_2 = 0.99$, and l_2 weight regularization with trade-off $\gamma = 0.0004$. Furthermore, extensive dataset augmentation is applied to prevent model overfitting. We refer to the supplementary material for more details.

Our Final Model. We analyze various optimization options and chose "our-4light-sepref" as our final model for desktop systems as it provides the best trade-off between accuracy vs. run-time. As depicted in Fig. 6a, our method improves the run-time performance of PWC-Net from 30 FPS to 85 FPS – a speed-up of factor 2.8. For Sintel training data, the accuracy drops by ≈ 0.5 px in EPE terms, however for test data the drop in accuracy is significant where the

Table 3: Runtime performance in milliseconds per frame. We measure the total processing time (without disk IO) and the individual stages on two GPU models (Nvidia GTX 1080Ti and RTX 3090). †Fast preset. Downsamples flow computation by $2\times$ and only uses 50 iterations of stabilization instead of 150.

Task ↓ Res. / GPU	Optical flow		Stabilization		Total	
	1080Ti	3090	1080Ti	3090	1080Ti	3090
1920×1080 px	66.8	40.0	184.1	42.7	250.8	82.7
1280×720 px	31.3	19.7	86.5	21.1	117.8	40.8
640×480 px	12.6	6.2	20.6	6.3	33.2	12.5
1920×1080 px†	26.2	13.0	71.1	16.5	97.3	29.5

final EPE is 7.43, see Fig. 5. Nevertheless, the accuracy is sufficient enough for enforcing warping-based consistency. To validate our design decisions, we conduct an extensive ablation study in which we vary the architectural and training choices – please see the supplementary for details. Furthermore, we tune our architecture for optical flow calculation on mobile devices using channel pruning and quantization, which we also detail in the supplementary material. Here, we improve run-time performance from 2.8 FPS to 24 FPS (iPad Pro 2020), and 1.5 FPS to 13 FPS (iPad Air) – an improvement of factor 8. Next to showing the general applicability of optical flow CNNs on mobile devices, this demonstrates that real-time on-device stabilization of videos using our presented approach will become feasible with a further moderate increase in mobile GPU computing power. A fast optical-flow-based warping enables our framework to interactively control the degree of consistency and generate visually plausible results.

4. Experimental Results

4.1. Implementation Details

All our experiments were performed on a consumer PC with an AMD Ryzen 1920X 12-Core CPU, 48 GB of RAM, and a Nvidia GTX 1080Ti and RTX 3090 graphics cards with VRAMs of 11

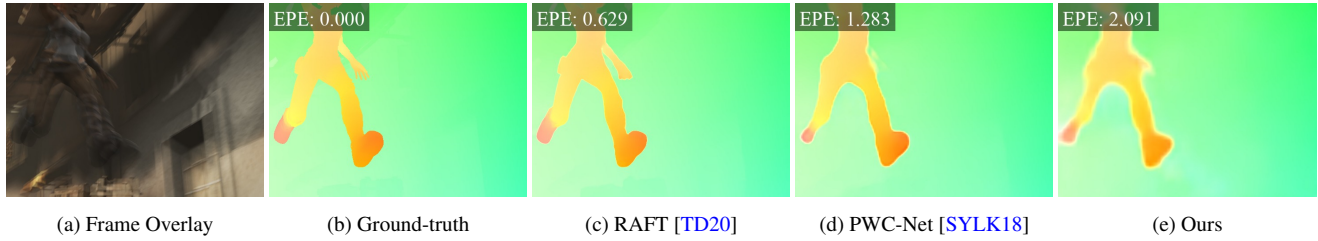


Figure 7: Optical flow estimated using the synthetic Sintel dataset [BWSB12].

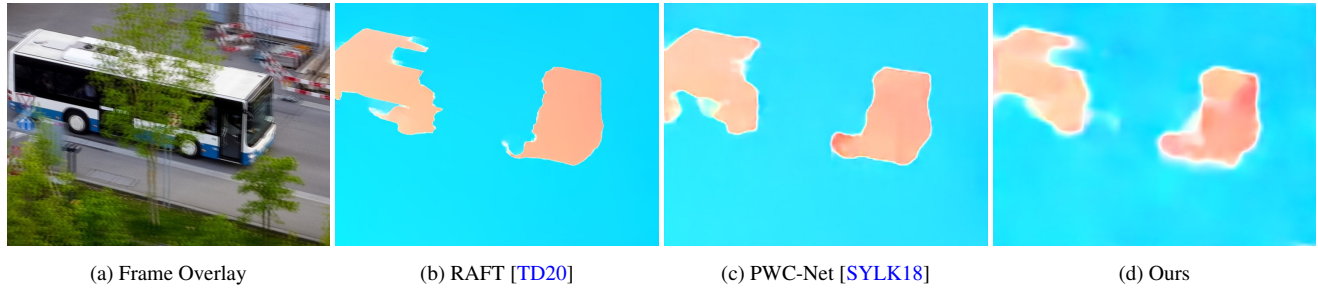


Figure 8: Optical flow estimated for the real-world dataset DAVIS [PTPC*17].

GB and 24 GB respectively. We implement a real-time video-consistency framework in C++, using ONNXRuntime for cross-platform acceleration of our lite optical-flow network, and implement the stabilization code using Nvidia CUDA (v11). In Tab. 3, we measure the runtime performance of our system. We find that an incoming stream of frames can be stabilized at real-time performance for VGA resolution even on low- and mid-tier GPUs and higher-tier GPUs (such as a RTX 3090) can stabilize HD at common video frame rates (approx. 24 FPS) and full-HD resolutions at interactive frame rates (> 10 FPS) (Tab. 3). We also test a fast preset that uses less iterations and computes optical flow on half-sized inputs, and find that a full-HD video stream can be processed in real-time at the cost of minor additional flickering - see the supplementary video for a comparison. We implement a graphical user interface that allows for real-time decoding and stabilization of stylized video streams, where the stabilization parameters can be interactively controlled, see the supplementary video for a demonstration.

4.2. Parameter Settings

Initially, we tune the parameters of our consistency framework towards achieving a low warping error (Tab. 5). We refer to this setting as *Ours-objective* with the following parameter values $k_1 = k_2 = 0.3$, $\alpha = 10 \times 10^3$, and $\lambda = 0.7$. However, we observed that even though the warping error indicated good temporal stability, subjective flickering, and artifacts were noticeable. Unlike existing approaches, our framework allows for interactive parameter adjustment. Thus, a parameter set that subjectively produces well-stabilized results on a broad range of tasks and videos was obtained experimentally. As our final version, we use the values of $k_1 = 0.3$, $k_2 = 0.5$, $\alpha = 6.5 \times 10^3$, and $\lambda = 2.0$ to generate all the images in the paper and the videos provided in the supplementary. We further compare *Ours-objective* settings with our final version as part

of our user study to validate our parameter choices. The consistent outputs obtained using the above parameter settings are compared against state-of-the-art approaches thereby showcasing its efficacy.

4.3. Optical Flow Results

We visualize optical flow on frames from the Sintel [BWSB12] dataset in Fig. 7 and compare it to state-of-the-art methods. All depicted methods have been fine-tuned on Sintel. We find that our optimized method has more blurry motion boundaries and misses estimating certain details accurately (e.g., the right hand, however, PWCNet also fails at this), but still captures the overall motion direction of objects correctly with a smooth flow field. Fig. 8 shows results for real-world videos on the DAVIS dataset [PTPC*17] (no ground-truth flow available). We find that some real-world image phenomena, such as complex/ambiguous occlusions (e.g., bus behind the tree) are not well-handled by state-of-the-art methods like RAFT [TD20] or PWC-Net [SYLK18], similarly, such results are also degraded for our optimized method. Besides the stronger blurred motion boundaries, we find that our network generally performs well and is also robust for real-world videos.

4.4. Consistent Outputs

We use videos from DAVIS [PPTM*16] dataset and other open-source videos (taken from [Vid] and [Pex]) for comparison. For per-frame stylization, we employ the following stylization techniques: Fast NST [JAFF16], WCT [LFY*17], and CycleGAN [ZPIE17]. The results for the method of Lai et al. and Bonneel et al. on videos taken from DAVIS [PPTM*16] and Videvo ([Vid]) are borrowed from the *results-dataset* provided by Lai et al. . For other videos, we employ the source code provided by the authors to generate the results. We compare our consistent outputs with that of Bonneel et al. [BTS*15] and Lai et al. [LHW*18] in Fig. 9. Among

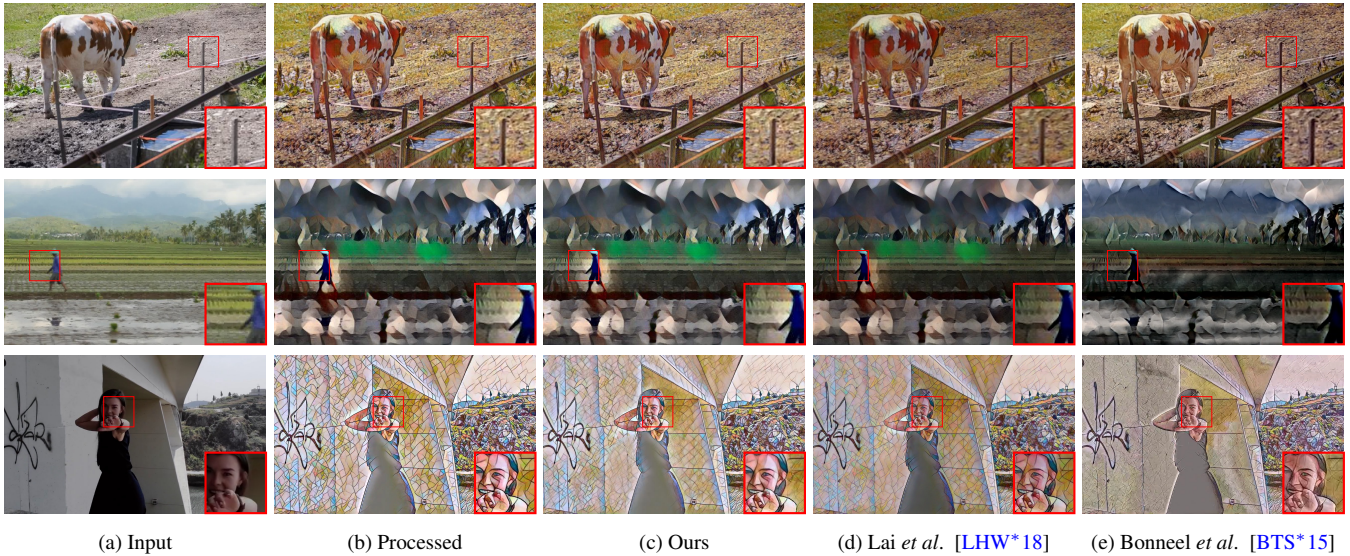


Figure 9: Comparing our results with Lai et al. [LHW*18] and Bonneel et al. [BTS*15] for three different video sequences. Note how the consistent output for Lai et al. and Bonneel et al. look different from the corresponding per-frame processed results.

the three competing methods Bonneel et al. is the least effective in preserving the underlying style for the final output (compare the second column with the fifth one in Fig. 9). Hyper-parameter tuning in the above method (with only global consistency) can provide a certain degree of consistency control. However, by employing both global and local consistency we achieve finer consistency control while being similar to the per-frame-processed result. For the method of Lai et al. , we observe some color bleeding or darkening in the output frames (compare the second column with the fourth one in Fig. 9). In comparison, we are able to preserve the style, color, and textures, while being consistent.

5. Evaluation

5.1. Quantitative Evaluation

Following Lai et al. [LHW*18], we measure the similarity between per-frame processed output and stabilized results, and the temporal warping error between consecutive stabilized frames.

For the former, we report the similarity in the form of the SSIM metric in Tab. 4. We achieve significantly higher similarity scores than the methods of Bonneel et al. [BTS*15] and Lai et al. [LHW*18]. Following [BTS*15] and [LHW*18], we also measure the temporal warping error between a frame V_t and the warped consecutive frame \hat{V}_{t+1} , defined as:

$$E_{\text{warp}}(V_t, V_{t+1}) = \frac{1}{\sum_{i=1}^N M_t^{(i)}} \sum_{i=1}^N M_t^{(i)} \|V_t^{(i)} - \hat{V}_{t+1}^{(i)}\|_1, \quad (9)$$

where $M_t \in \{0, 1\}$ is a non-occlusion mask [LHW*18, RDB18], indicating non-occluded regions. The warped frame \hat{V}_{t+1} is obtained by calculating the optical flow (using GMA [JCL*21]) between frames V_t, V_{t+1} , and applying a backwards warping to frame V_{t+1} . We compute E_{warp} for every frame of a video and then averaged to obtain the warping error of a video $E_{\text{warp}}(V)$. In Tab. 5 we report the

average warping error per dataset (see the supplementary for a per-task breakdown). We find that the warping error is slightly higher than that of Bonneel et al. [BTS*15] and Lai et al. [LHW*18]. However, as Lai et al. [LHW*18] notes, results with high temporal stability (expressed by a low warping error) can also be achieved via temporally smoothing the video, which can be seen in various results of Bonneel et al. [BTS*15]. Our qualitative results in the form of a user study Sec. 5.2 further substantiate the divide between warping error (as a stability metric) and perceived stability.

Using other Optical Flow Computations. We also tested other optical flow methods within our pipeline which were either faster [KTDVG16] or more accurate [TD20]. For the fast optical method by Kroeger et al. [KTDVG16](DIS) the final output is less consistent than ours in both objective and subjective metrics. Using DIS for our stabilization, the average warp-error is higher (Tab. 5), and perceptual-similarity with the per-frame processed result is lower than ours (0.9 in SSIM over DAVIS and VIDEVO). Visually, DIS-stabilized results show significantly more flickering, validating our design choice for the optical flow. A much more accurate optical flow is given by the method of Teed et al. [TD20] (RAFT) at the cost of slow computation. The stabilized results obtained using RAFT look visually indistinguishable from the one obtained using our flow; the average warp-error is the same or marginally lower (Tab. 5), while the perceptual-similarity is the same in terms of SSIM as in Tab. 4. Due to visually unnoticeable and metric-wise only minor differences for RAFT, we conjecture that there will not be any significant improvement in output quality for even more accurate flow methods.

5.2. Qualitative Evaluation

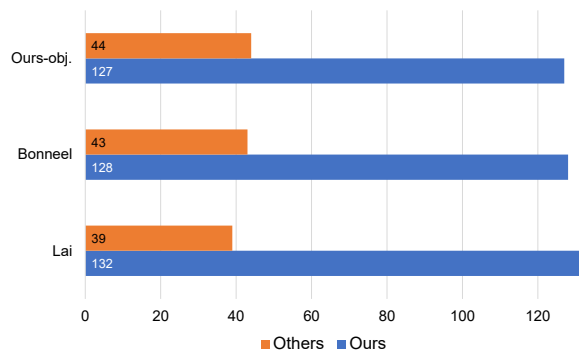
For qualitative evaluation, we perform a subjective user study where we ask participants to compare the temporally-consistent re-

Table 4: Quantitative evaluation on perceptual distance using SSIM (higher = more similar to per-frame processed result).

Task	DAVIS			VIDEVO		
	[BTS*15]	[LHW*18]	Ours	[BTS*15]	[LHW*18]	Ours
CycleGAN/photo2ukiyoe [ZPIE17]	0.693	0.781	0.978	0.626	0.743	0.980
CycleGAN/photo2vangogh [ZPIE17]	0.707	0.792	0.961	0.679	0.789	0.965
fast-neural-style/rain-princess [JAFF16]	0.553	0.799	0.921	0.491	0.796	0.920
fast-neural-style/udnie [JAFF16]	0.597	0.785	0.956	0.579	0.747	0.959
WCT/antimonocromatismo [LFY*17]	0.389	0.811	0.915	0.388	0.761	0.914
WCT/asheville [LFY*17]	0.329	0.801	0.904	0.348	0.771	0.901
WCT/candy [LFY*17]	0.289	0.763	0.882	0.310	0.738	0.885
WCT/feathers [LFY*17]	0.418	0.863	0.891	0.415	0.848	0.888
WCT/sketch [LFY*17]	0.370	0.845	0.923	0.370	0.833	0.922
WCT/wave [LFY*17]	0.358	0.700	0.902	0.352	0.637	0.899
Average	0.470	0.794	0.923	0.456	0.766	0.923

Table 5: Flow warping error average over tasks shown in Tab. 4. A per-task breakdown is shown in the supplementary. Note that the slightly higher warping error (lower is better) of our method is subjectively not noticeable as we show in a user study.

Dataset	V_p	[BTS*15]	[LHW*18]	Ours	Ours-RAFT	Ours-DIS
DAVIS	0.056	0.034	0.040	0.046	0.045	0.050
VIDEVO	0.051	0.036	0.036	0.042	0.042	0.046

**Figure 10:** Statistics of the user study results on removal of temporal flickering from per-frame stylized videos. For 19 participants and 9 different videos we compare our method against Bonneel et al., Lai et al., and Ours-objective through a total of 171 randomized A/B tests.

result obtained using our method with that of Lai et al. [LHW*18], Bonneel et al. [BTS*15], and Ours-objective – a different parameter setting of ours. We use 9 different videos for this purpose: 3 from DAVIS [PPTM*16], 3 from Videvo [Vid], and 3 from Pexels [Pex] datasets respectively. For each of the above videos we stylize them using either the Fast NST [JAFF16] (in the styles of *udnie*, *rain-princess*, and *mosaic*) or WCT [LFY*17] (in the styles of *wave* and *antimono*) or Cycle-GAN (in the styles of *photo2vangogh* and *photo2ukiyoe*). For each sample, we show the input video and its per-frame stylized version on the top row of the user-study interface for inference. In the bottom row, we show two different versions of the temporally stabilized output where one of them is ours. We ask the participants to select the output which best preserves:

(i) temporal consistency and (ii) similarity with the per-frame processed video. For 9 videos and 3 other competing methods, each user sees a total of 27 blind A/B tests which are shown in a randomized order to each participant. In total, 19 persons (3 female and 16 male) between the ages of 22 to 43 years participated in the study. Fig. 10 shows that our method surpasses all others by a large margin. It was interesting to observe that for certain cases the method of Bonneel et al. which degrades the processed style significantly was still preferred by users over others due to its high consistency quality. We also show subjective user preference for our method over the methods of Shekhar et al. [SST*19] and Thimonier et al. [TDKP21] via another user study in the supplementary material. Shekhar et al. which is completely based on temporal denoising is prone to introduce motion blur artifacts and is less efficient in terms of enforcing global consistency. Similar to Lai et al., Thimonier et al. introduces artifacts in the form of color bleeding and darkening, see the supplementary video.

5.3. Comparison to Other Methods

Keyframe-based Stylization (KBS). The goal of KBS is to propagate the style from a few selected keyframes to the rest of the sequence, usually with per-sequence pretraining, running in an offline scenario. Our focus on the other hand is blind video consistency, i.e., we assume to have no control over the per-frame stylization process and consider a video stream as our application scenario. However, some KBS methods such as Texler et al. [TFK*20] or Futschik et al. [FKL*21] are capable of running in online scenarios, under the assumption that the video stream is similar enough to the pre-trained frames (e.g., a webcam stream). The results however often exhibit temporal flickering. To combat this, Texler et al. propose a method for temporal stabilization that involves pre-filtering frames using a motion-compensated bilateral filter and encoding location data using a mixture of Gaussians. In Fig. 11 and the



Figure 11: Comparing temporal stabilization for keyframe-based stylization (KBS) [TFK*20]. We compare the stabilization approach proposed by Texler et al. [TFK*20] to our method. To obtain the unstabilized KBS video, we select three keyframes (1st, middle, last) from a video from DAVIS, apply for style transfer, and train a KBS model [TFK*20] on them, which is then used to stylize the remaining the frames.

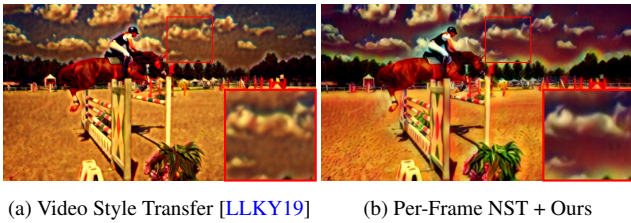


Figure 12: Comparing to video style transfer [LLKY19]. We compare the implicit stabilization of their video style transfer technique to their per-frame NST stabilized with our approach.

supplementary video, we compare their stabilization approach to ours on KBS-stylized videos. For videos stylized using NST, visual similarities are apparent between both stabilized versions, though our method displays superior detail preservation. However, for complex-structured styles that introduce texture into homogeneous regions, such as pencil drawings, their Gaussian-mixture-based stabilization improves texture adherence. In contrast, our method may lead to over-smoothing due to inaccurate flow computation in such featureless regions (see supplementary video). This effect can be mitigated to a certain degree by employing a lower temporal consistency factor (λ). Compared to Gaussian-mixture-based stabilization our approach runs at least an order of magnitude faster, making it better suited for interactive scenarios such as KBS-stylizing and stabilizing an incoming video stream. Futschik et al. improves on the temporal consistency of Texler et al. by considering additional frames from the video during training. However, this makes it less applicable to out-of-domain videos (i.e., content not seen during training) which is common in video streams. Our method can also effectively stabilize such videos as shown in the supplementary video.

Video Style Transfer. In Fig. 12, we compare our method to the arbitrary style transfer for videos of Li et al. [LLKY19]. Despite their method having full control over the stylization process, their results exhibit more temporal flickering and blurring, particularly in smooth regions such as the sky. Their video style transfer method also tends to under-stylize image features compared to their

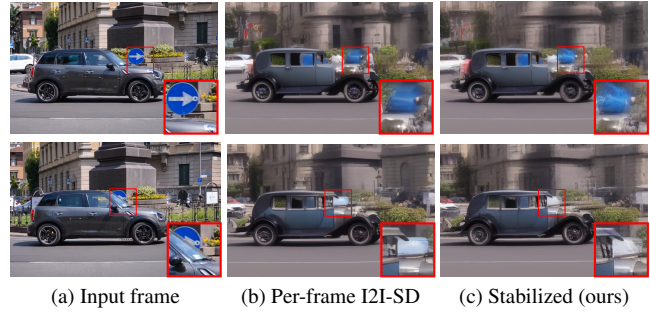


Figure 13: Results on per-frame img2img stable diffusion (I2I-SD) [RBL*22] applied with the prompt "a 1920s car in a roundabout". Latent codes are interpolated with the previous frames to improve consistency. We use CfG scale = 7.5 and a denoising factor of 0.4.

per-frame style transfer. Please see the supplementary video for a video-based comparison.

6. Discussion

Our approach takes a video pair as an input: (i) the original and (ii) its per-frame stylized version. We assume that the stylization is based on the input image gradients and appears as variations in the form of colors and/or textures. Thereby, we employ the original video as a guide for enforcing consistency. However, for text-guided generative arts such as recent diffusion model-based approaches [RDN*22, RBL*22] the stylized frames are often only weakly correlated with the original input, and we cannot handle such cases. In Fig. 13 we provide an example of a per-frame stylization using stable diffusion [RBL*22], in which despite using a latent pre-initialization from previous frames, new details are hallucinated in every frame, which cannot be effectively removed by our method, resulting in a blurry output video.

For the evaluation, we mainly use CNN-based stylization techniques. However, our approach can also handle classical stylization algorithms [KCWI13], we show a few such examples in the supplementary. Our local consistency component comprising a convex combination of temporal neighbors can be seen as a crude form of local temporal denoising. Previously it has been shown that temporal denoising is effective in enforcing consistency [SST*19]. We conjecture that efficient temporal denoising combined with flow-based warping can further improve temporal stabilization not only for stylization but also for other tasks. We show examples for such non-stylization tasks, particularly for image enhancement (DBL [GCB*17]) and intrinsic decomposition, in the supplementary.

We start with the assumption that temporal flickering is not completely undesirable for the task of stylization and thus we provide interactive consistency control. However, during the subjective user study, we observed that participants had different tolerance levels for flickering in the foreground as compared to that in the background. As part of future work, one can use depth-based or saliency-based masks to vary the consistency control parameters spatially for a more visually pleasing result.

Limitation. Our approach tends to have ghosting artifacts for fast-moving objects where the object motion between consecutive frames is large (Fig. 14). The above can be reduced by reducing the upper limit of w_p (i.e., k_1), however, such a reduction also reduces consistency in the final output. We argue that since we provide interactive control of parameters the above trade-off between artifacts vs. consistency will not significantly hinder its usability.

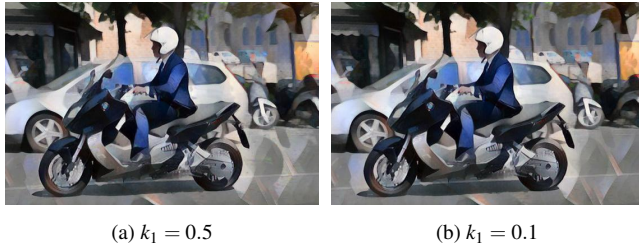


Figure 14: The ghosting artifacts on the rear wheel of the scooter are significant in the final output for $k_1 = 0.5$, however, they are significantly reduced for $k_1 = 0.1$.

7. Conclusions

We propose an approach that makes per-frame stylized videos temporally coherent irrespective of the underlying stylization applied to individual frames. At this, we introduce a novel temporal consistency term that combines local and global consistency aspects. We maintain similarity with the per-frame processed result by minimizing the difference in the gradient domain. Unlike previous approaches, we provide interactive consistency control by computing optical flow on the incoming video stream at high speed and with sufficient accuracy for stabilization. The fast optical-flow inference is achieved by developing a lightweight flow network architecture based on PWC-Net. The entire optimization solving is GPU-based and runs at real-time frame rates for HD resolution. We showcase that our temporally consistent output is preferred over the output of competing methods by conducting a user study. As part of future work, we would like to employ learning-based temporal denoising to further improve the quality of results. Moreover, we would like to explore the usage of depth-based and saliency-based masks to spatially vary consistency parameters according to perceptual principles. We hope that our design paradigm of interactive consistency control will potentially make per-frame video stylization more user-friendly.

Acknowledgements

We thank the anonymous reviewers for their valuable feedback. We thank Jobin Idiculla Wattasseril for helping with the user study. This work was funded by the German Federal Ministry of Education and Research (BMBF) (through grants 01IS15041 – “md-ViProject” and 01IS19006 – “KI-Labor ITSE”) and the Research School on “Service-Oriented Systems Engineering” of the Hasso Plattner Institute.

References

[App] APPLE: Coreml. URL: <https://developer.apple.com/machine-learning/core-ml>. 17

- [BCCZ08] BHAT P., CURLESS B., COHEN M., ZITNICK C. L.: Fourier analysis of the 2d screened poisson equation for gradient domain problems. In *Proceedings of the 10th European Conference on Computer Vision: Part II* (2008), ECCV '08, Springer-Verlag, p. 114–128. doi: [10.1007/978-3-540-88688-4_9](https://doi.org/10.1007/978-3-540-88688-4_9). 4
- [BCK*13] BÉNARD P., COLE F., KASS M., MORDATCH I., HEGARTY J., SENN M. S., FLEISCHER K., PESARE D., BREEDEN K.: Stylizing animation by example. *ACM Trans. Graph.* 32, 4 (jul 2013). doi: [10.1145/2461912.2461929](https://doi.org/10.1145/2461912.2461929). 1, 4
- [BLV*10] BÉNARD P., LAGAE A., VANGORP P., LEFEBVRE S., DRETTAKIS G., THOLLOT J.: A dynamic noise primitive for coherent stylization. *Computer Graphics Forum* 29, 4 (2010), 1497–1506. doi: <https://doi.org/10.1111/j.1467-8659.2010.01747.x>. 1
- [BNTS07] BOUSSEAU A., NEYRET F., THOLLOT J., SALESIN D.: Video watercolorization using bidirectional texture advection. *ACM Trans. Graph.* 26, 3 (jul 2007), 104–es. doi: [10.1145/1276377.1276507](https://doi.org/10.1145/1276377.1276507). 1, 3
- [BSFG09] BARNES C., SHECHTMAN E., FINKELSTEIN A., GOLDMAN D. B.: Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.* 28, 3 (jul 2009). doi: [10.1145/1531326.1531330](https://doi.org/10.1145/1531326.1531330). 3
- [BTC13] BÉNARD P., THOLLOT J., COLLOMOSSE J.: *Temporally Coherent Video Stylization*. Springer, 2013, pp. 257–284. doi: [10.1007/978-1-4471-4519-6_13](https://doi.org/10.1007/978-1-4471-4519-6_13). 1, 4
- [BTS*15] BONNEEL N., TOMPKIN J., SUNKAVALLI K., SUN D., PARIS S., PFISTER H.: Blind video temporal consistency. *ACM Trans. Graph.* 34, 6 (oct 2015). doi: [10.1145/2816795.2818107](https://doi.org/10.1145/2816795.2818107). 1, 2, 3, 4, 5, 8, 9, 10, 19
- [BWSB12] BUTLER D. J., WULFF J., STANLEY G. B., BLACK M. J.: A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision (ECCV)* (2012), pp. 611–625. doi: [10.1007/978-3-642-33783-3_44](https://doi.org/10.1007/978-3-642-33783-3_44). 6, 7, 8, 15, 17, 18, 20
- [BZCC10] BHAT P., ZITNICK C. L., COHEN M., CURLESS B.: Gradientshop: A gradient-domain optimization framework for image and video filtering. *ACM Trans. Graph.* 29, 2 (2010). doi: [10.1145/1731047.1731048](https://doi.org/10.1145/1731047.1731048). 4
- [CLY*17] CHEN D., LIAO J., YUAN L., YU N., HUA G.: Coherent online video style transfer. In *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 1114–1123. doi: [10.1109/ICCV.2017.126](https://doi.org/10.1109/ICCV.2017.126). 4
- [DBZY15] DONG X., BONEV B., ZHU Y., YUILLE A. L.: Region-based temporally consistent video post-processing. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 714–722. doi: [10.1109/CVPR.2015.7298671](https://doi.org/10.1109/CVPR.2015.7298671). 3
- [DTD*21] DENG Y., TANG F., DONG W., HUANG H., MA C., XU C.: Arbitrary video style transfer via multi-channel correlation. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 2 (May 2021), 1210–1217. URL: <https://ojs.aaai.org/index.php/AAAI/article/view/16208>. 4
- [Fan19] FANG G.: Torch-pruning: A pytorch pruning toolkit for structured neural network pruning and automatic layer dependency maintaining., 2019. URL: <https://github.com/VainF/Torch-Pruning>. 15, 17
- [FDI*15] FISCHER P., DOSOVITSKIY A., ILG E., HÄUSSER P., HAZIRBAS C., GOLKOV V., VAN DER SMAGT P., CREMERS D., BROX T.: Flownet: Learning optical flow with convolutional networks. In *International Conference on Computer Vision (ICCV)* (2015), p. 2758–2766. doi: [10.1109/ICCV.2015.316](https://doi.org/10.1109/ICCV.2015.316). 7, 15, 17, 18
- [FKL*21] FUTSCHIK D., KUČERA M., LUKÁČ M., WANG Z., SHECHTMAN E., ŠYKORA D.: Stalp: Style transfer with auxiliary limited pairing. *Computer Graphics Forum* 40, 2 (2021), 563–573. doi: <https://doi.org/10.1111/cg.f.142655>. 1, 4, 10
- [FLJ*14] FIŠER J., LUKÁČ M., JAMRIŠKA O., ČADÍK M., GINGOLD

- Y., ASENTE P., SÝKORA D.: Color me noisy: Example-based rendering of hand-colored animations with temporal noise control. In *Proceedings of the 25th Eurographics Symposium on Rendering* (2014), EGSR '14, Eurographics Association, p. 1–10. [2, 4](#)
- [GCB*17] GHARBI M., CHEN J., BARRON J. T., HASINOFF S. W., DURAND F.: Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 118. [11](#)
- [GEB16] GATYS L. A., ECKER A. S., BETHGE M.: Image style transfer using convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 2414–2423. [doi:10.1109/CVPR.2016.265.4](#)
- [GJAF17] GUPTA A., JOHNSON J., ALAHI A., FEI-FEI L.: Characterizing and improving stability in neural style transfer. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22–29, 2017* (2017), pp. 4087–4096. [doi:10.1109/ICCV.2017.438.4](#)
- [Hae90] HAEBERLI P.: Paint by numbers: Abstract image representations. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques* (1990), SIGGRAPH '90, Association for Computing Machinery, p. 207–214. [doi:10.1145/97879.97902.1](#)
- [HBP*20] HOFINGER M., BULÒ S. R., PORZI L., KNAPITSCH A., POCK T., KONTSCHIEDER P.: Improving optical flow on a pyramid level. In *European Conference on Computer Vision (ECCV)* (2020), pp. 770–786. [doi:10.1007/978-3-030-58604-1_46.18](#)
- [HLvdMW17] HUANG G., LIU Z., VAN DER MAATEN L., WEINBERGER K. Q.: Densely connected convolutional networks. In *Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 2261–2269. [doi:10.1109/CVPR.2017.243.7,15](#)
- [HTL18] HUI T.-W., TANG X., LOY C. C.: Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *Computer Vision and Pattern Recognition (CVPR)* (2018), pp. 8981–8989. [doi:10.1109/CVPR.2018.00936.3,7,17](#)
- [HTL20] HUI T. W., TANG X., LOY C. C.: A lightweight optical flow cnn - revisiting data fidelity and regularization. In *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2020). [doi:10.1109/TPAMI.2020.2976928.3,6,7,15,17](#)
- [HWL*17] HUANG H., WANG H., LUO W., MA L., JIANG W., ZHU X., LI Z., LIU W.: Real-time neural style transfer for videos. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 7044–7052. [doi:10.1109/CVPR.2017.745.4](#)
- [HZC*17] HOWARD A. G., ZHU M., CHEN B., KALENICHENKO D., WANG W., WEYAND T., ANDREETTO M., ADAM H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. In *CoRR* (2017). [arXiv:1704.04861.7,15,18](#)
- [IMS*17] ILG E., MAYER N., SAIKIA T., KEUPER M., DOSOVITSKIY A., BROX T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 1647–1655. [doi:10.1109/CVPR.2017.179.3,6,18](#)
- [JAFF16] JOHNSON J., ALAHI A., FEI-FEI L.: Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision – ECCV 2016* (2016), pp. 694–711. [doi:10.1007/978-3-319-46475-6_43.8,10](#)
- [JCL*21] JIANG S., CAMPBELL D., LU Y., LI H., HARTLEY R.: Learning to estimate hidden motions with global motion aggregation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 9772–9781. [2,3,6,9,19](#)
- [JST*19] JAMRIŠKA O., SOCHOROVÁ V., TEXLER O., LUKÁČ M., FIŠER J., LU J., SHECHTMAN E., SÝKORA D.: Stylizing video by example. *ACM Trans. Graph.* 38, 4 (jul 2019). [doi:10.1145/3306346.3323006.1,4](#)
- [JYF*20] JING Y., YANG Y., FENG Z., YE J., YU Y., SONG M.: Neural style transfer: A review. *IEEE Transactions on Visualization and Computer Graphics* 26, 11 (2020), 3365–3385. [doi:10.1109/TVCG.2019.2921336.1](#)
- [KB15] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)* (2015). [doi:10.1145/1.505367.17](#)
- [KCWI13] KYPRIANIDIS J. E., COLLOMOSSE J., WANG T., ISENBERG T.: State of the "art": A taxonomy of artistic stylization techniques for images and video. *IEEE Transactions on Visualization and Computer Graphics* 19, 5 (2013), 866–885. [doi:10.1109/TVCG.2012.160.1,11](#)
- [KP11] KASS M., PESARE D.: Coherent noise for non-photorealistic rendering. *ACM Trans. Graph.* 30, 4 (jul 2011). [doi:10.1145/2010324.1964925.1](#)
- [KTDVG16] KROEGER T., TIMOFTE R., DAI D., VAN GOOL L.: Fast optical flow using dense inverse search. In *Computer Vision – ECCV 2016* (2016), Leibe B., Matas J., Sebe N., Welling M., (Eds.), pp. 471–488. [9](#)
- [LFY*17] LI Y., FANG C., YANG J., WANG Z., LU X., YANG M.-H.: Universal style transfer via feature transforms. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (2017), NIPS'17, p. 385–395. URL: <https://dl.acm.org/doi/10.5555/3294771.3294808>. [8,10](#)
- [LH19] LOSHCHELOV I., HUTTER F.: Fixing weight decay regularization in adam. In *International Conference on Learning Representations (ICLR)* (2019). URL: <https://openreview.net/forum?id=rk6qdGgCZ.7,17>
- [LHW*18] LAI W.-S., HUANG J.-B., WANG O., SHECHTMAN E., YUMER E., YANG M.-H.: Learning blind video temporal consistency. In *Computer Vision – ECCV 2018* (2018), pp. 179–195. [1,2,3,5,8,9,10,18,19](#)
- [Lit97] LITWINOWICZ P.: Processing images and video for an impressionist effect. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., p. 407–414. [doi:10.1145/258734.258893.3](#)
- [LKD*16] LI H., KADAV A., DURDANOVIC I., SAMET H., GRAF H. P.: Pruning filters for efficient convnets. In *International Conference on Learning Representations (ICLR)* (2016). URL: <https://openreview.net/forum?id=rJqFGTslg.15,17>
- [LLKY19] LI X., LIU S., KAUTZ J., YANG M.: Learning linear transformations for fast image and video style transfer. In *Proceedings - 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2019* (June 2019), pp. 3804–3812. [doi:10.1109/CVPR.2019.00393.4,11](#)
- [LWA*12] LANG M., WANG O., AYDIN T., SMOLIC A., GROSS M.: Practical temporal consistency for image-based graphics applications. *ACM Trans. Graph.* 31, 4 (jul 2012). [doi:10.1145/2185520.2185530.2](#)
- [LZH*20] LIU L., ZHANG J., HE R., LIU Y., WANG Y., TAI Y., LUO D., WANG C., LI J., HUANG F.: Learning by analogy: Reliable supervision from transformations for unsupervised optical flow estimation. In *Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 6488–6497. [doi:10.1109/CVPR42600.2020.00652.3,6,7,15](#)
- [MIH*16] MAYER N., ILG E., HÄUSSER P., FISCHER P., CREMERS D., DOSOVITSKIY A., BROX T.: A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 4040–4048. [doi:10.1109/CVPR.2016.438.7,17,18](#)
- [Nik18] NIKLAUS S.: pytorch-pwc: a reimplementation of pwc-net in pytorch that matches the official caffe version, 2018. URL: <https://github.com/sniklaus/pytorch-pwc.7>
- [NSC*11] NORIS G., SÝKORA D., COROS S., WHITED B., SIMMONS M., HORNING A., GROSS M., SUMNER R. W.: Temporal noise control for sketchy animation. In *NPAC '11* (2011), Association for Computing Machinery, p. 93–98. [doi:10.1145/2024676.2024691.1,3](#)

- [P*19] PASZKE A., ET AL.: Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NIPS)*. 2019, pp. 8024–8035. 7, 17
- [Pex] PEXELS: Pexels. URL: <https://www.pexels.com/>. 8, 10
- [PP19] PUY G., PÉREZ P.: A flexible convolutional solver for fast style transfers. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 8955–8964. doi:10.1109/CVPR.2019.00917. 4
- [PPTM*16] PERAZZI F., PONT-TUSET J., MCWILLIAMS B., VAN GOOL L., GROSS M., SORKINE-HORNUNG A.: A benchmark dataset and evaluation methodology for video object segmentation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 724–732. doi:10.1109/CVPR.2016.85. 8, 10
- [PTPC*17] PONT-TUSET J., PERAZZI F., CAELLES S., ARBELAEZ P., SORKINE-HORNUNG A., GOOL L. V.: The 2017 DAVIS challenge on video object segmentation. In *CoRR* (2017). arXiv:1704.00675. 8, 21
- [Qia99] QIAN N.: On the momentum term in gradient descent learning algorithms. *Neural Networks* 12, 1 (1999), 145–151. doi:[https://doi.org/10.1016/S0893-6080\(98\)00116-6](https://doi.org/10.1016/S0893-6080(98)00116-6)
- [RB17] RANJAN A., BLACK M. J.: Optical flow estimation using a spatial pyramid network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 2720–2729. doi:10.1109/CVPR.2017.291. 6
- [RBL*22] ROMBACH R., BLATTMANN A., LORENZ D., ESSER P., OMMER B.: High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 10684–10695. doi:10.1109/CVPR52688.2022.01042. 2, 11
- [RDB18] RUDER M., DOSOVITSKIY A., BROX T.: Artistic style transfer for videos and spherical images. *International Journal of Computer Vision* 126, 11 (Nov 2018), 1199–1219. doi:10.1007/s11263-018-1089-z. 4, 9
- [RDN*22] RAMESH A., DHARIWAL P., NICHOL A., CHU C., CHEN M.: Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125* (2022). 11
- [SHR*22] SUN D., HERRMANN C., REDA F., RUBINSTEIN M., FLEET D. J., FREEMAN W. T.: Disentangling architecture and training for optical flow. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII* (2022), Springer-Verlag, p. 165–182. doi:10.1007/978-3-031-20047-2_10. 6
- [SID17] SEMMO A., ISENBERG T., DÖLLNER J.: Neural style transfer: A paradigm shift for image-based artistic rendering? In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering* (2017), NPAR '17, Association for Computing Machinery. doi:10.1145/3092919.3092920. 1
- [SST*19] SHEKHAR S., SEMMO A., TRAPP M., TURSUN O., PASEWALDT S., MYSZKOWSKI K., DÖLLNER J.: Consistent Filtering of Videos and Dense Light-Fields Without Optic-Flow. In *Vision, Modeling and Visualization* (2019), Schulz H.-J., Teschner M., Wimmer M., (Eds.). doi:10.2312/vmv.20191326. 2, 3, 4, 5, 10, 11, 15, 18
- [SVH*21] SUN D., VLASIC D., HERRMANN C., JAMPANI V., KRAININ M., CHANG H., ZABIH R., FREEMAN W. T., LIU C.: Autoflow: Learning a better training set for optical flow. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 10088–10097. doi:10.1109/CVPR46437.2021.00996. 6
- [SYLK18] SUN D., YANG X., LIU M.-Y., KAUTZ J.: PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *Computer Vision and Pattern Recognition (CVPR)* (2018), pp. 8934–8943. doi:10.1109/CVPR.2018.00931. 3, 6, 7, 8, 17, 18, 20, 21
- [TD20] TEED Z., DENG J.: Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision – ECCV 2020* (2020), Vedaldi A., Bischof H., Brox T., Frahm J.-M., (Eds.), pp. 402–419. doi:10.1007/978-3-030-58536-5_24. 2, 3, 6, 8, 9, 18, 20, 21
- [TDKP21] THIMONIER H., DESPOIS J., KIPS R., PERROT M.: Learning long term style preserving blind video temporal consistency. In *2021 IEEE International Conference on Multimedia and Expo (ICME)* (2021), IEEE, pp. 1–6. 1, 2, 3, 10, 15, 18
- [TFK*20] TEXLER O., FUTSCHIK D., KUČERA M., JAMRIŠKA O., SOCHOROVÁ V., CHAI M., TULYAKOV S., SÝKORA D.: Interactive video stylization using few-shot patch-based training. *ACM Trans. Graph.* 39, 4 (2020). doi:10.1145/3386569.3392453. 1, 4, 10, 11
- [Vid] VIDEVO: Videvo. URL: <https://www.videvo.net/>. 8, 10
- [WOG06] WINNEMÖLLER H., OLSEN S. C., GOOCH B.: Real-time video abstraction. *ACM Trans. Graph.* 25, 3 (jul 2006), 1221–1226. doi:10.1145/1141911.1142018. 3
- [YCC17] YAO C.-H., CHANG C.-Y., CHIEN S.-Y.: Occlusion-aware video temporal consistency. In *Proceedings of the 25th ACM International Conference on Multimedia* (2017), MM '17, Association for Computing Machinery, p. 777–785. doi:10.1145/3123266.3123363. 1, 2, 3
- [YR19a] YANG G., RAMANAN D.: Volumetric correspondence networks for optical flow. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems* (Red Hook, NY, USA, 2019), Curran Associates Inc. URL: <https://dl.acm.org/doi/pdf/10.5555/3454287.3454359.6,7>
- [YR19b] YANG G., RAMANAN D.: Volumetric correspondence networks for optical flow. In *Advances in Neural Information Processing Systems (NIPS)* (2019), pp. 794–805. URL: <https://papers.nips.cc/paper/2019/hash/bbf94b34eb32268ada57a3be5062fe7d-Abstract.html>. 17
- [ZGWX05] ZHU S.-C., GUO C.-E., WANG Y., XU Z.: What are textons? *International Journal of Computer Vision* 62, 1 (Apr 2005), 121–143. doi:10.1023/B:VISI.0000046592.70770.61. 5
- [ZPIE17] ZHU J.-Y., PARK T., ISOLA P., EFROS A. A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 2242–2251. doi:10.1109/ICCV.2017.244. 8, 10

Supplementary Material

Some details had to be omitted from the main paper due to the page limit; we present those details here. In the following, we report on ablation experiments in Sec. 1.1, which were carried out to determine the best performing fast optical-flow network, and also expand on quantization and pruning which were employed for our mobile-optimized network. In Sec. 1.2 we expand on training and implementation details. In Sec. 2 we provide detailed numbers for the warping error. In Sec. 3 we compare our method subjectively against Shekhar *et al.* [SST*19] and Thimonier *et al.* [TDKP21] through a user study. Finally, in Sec. 4, we present further visual results of our optical-flow network.

1. Fast Optical Flow Network

1.1. Ablation Study

To analyze our optimization steps, we compare different variants of our CNN. All variants are trained on the full dataset schedule unless stated otherwise. We make use of Sintel Final Train dataset [BWSB12] as a benchmark and measure accuracy (in terms of EPE), number of parameters, and run-time for different variants. Due to different desktop and mobile GPU hardware, the run-time performance can vary between platforms, thus we measure them separately.

Table 6: Comparison of DenseNet [HLvdMW17] and light [LZH*20] connections on a desktop system and mobile devices on Sintel Final Train.

Variant	EPE (↓)	Params Million (M) (↓)	Desktop FPS (↑)	iPad Air FPS (↑)	iPad Pro FPS (↑)
dense	2.507	9.36	29.97	1.53	2.80
light	<u>2.825</u>	<u>5.99</u>	<u>40.26</u>	<u>2.83</u>	<u>5.09</u>

DenseNet Connection Replacement. As a first architectural improvement, we replace DenseNet [HLvdMW17] connections in the flow estimators with light connections [LZH*20]. Replacing these results in a significant run-time improvement on both desktop systems and mobile devices, with a larger relative speed-up on mobile devices (Tab. 6). We conjecture that convolutions with a large number of channels (dense architecture uses up to 565 channels) might perform worse on mobile GPUs due to smaller memory and cache sizes. Thus, reducing these high channel counts results in a larger speed-up on mobile devices. The light connections result in a loss in accuracy (Tab. 6), but due to the significant run-time improvements, we find it a reasonable trade-off and use light connections in the following experiments and in our proposed mobile architecture.

Channel Reduction. We reduce the number of channels throughout the CNN [HZC*17]. In this case, the loss in accuracy and achieved trade-off is not beneficial (Tab. 7). We hypothesize that channel reduction is potentially better for high-level Computer Vision (CV) tasks, where high-dimensional convolution features are mapped to very low-dimensional results [HZC*17]. Optical flow,

Table 7: Ablation of different channel reduction on desktop system

Variant	EPE (↓)	Params M (↓)	Desktop FPS (↑)	iPad Air FPS (↑)	iPad Pro FPS (↑)
<u>5 estimators</u>	<u>2.825</u>	<u>5.99</u>	<u>40.26</u>	<u>2.83</u>	<u>5.09</u>
-25% channels	3.659	3.55	46.86	3.95	6.65
-50% channels	4.236	1.73	56.07	6.33	10.92

however, requires pixel-precise predictions of continuous values (motion vectors) and thus requires a much higher spatial fidelity. Furthermore, we observe that the relative speed-up on mobile devices is again higher than on desktop systems which supports our previous belief that larger convolutions are more difficult for mobile GPUs with smaller memory and cache sizes.

Flow Estimators. We evaluate different configurations of separable convolutions for the five flow-estimator modules. Replacing all convolutions in the flow estimators with separable convolutions leads to a significant loss in accuracy (Tab. 8). The last two flow estimators operate on the highest pyramid resolutions and have the largest impact on run-time performance. Thus, loss of accuracy can be minimized by using separable convolutions only for the last two flow estimators. Moreover, we find that removing only the last flow estimator leads to a larger speed-up and overall better trade-off [HTL20], both on desktop systems and mobile devices (Tab. 8). The last flow estimator – operating on quarter input resolution – comprises only 11.3% of parameters, but removing it results in more than 100% speed-up on mobile devices.

Refinement. For the four previously chosen flow estimators, we find that dense refinement can be replaced by separable convolutions which even results in a slight increase in accuracy on both desktop and mobile devices (Tab. 9). For five flow estimators, we observe that dense refinement has a larger impact on accuracy.

Pruning: As an additional improvement for mobile deployment, we evaluate pruning as a post-training step. Convolution filter pruning is applied as proposed by Li *et al.* [LKD*16] and a l_1 strategy combined with automatic consistency checks [Fan19] is used for selecting which filters to prune. We apply it to each convolution layer that has more than two output channels. To keep pruning simple, we prune the same percentage of filters from each layer and then perform a single re-training to account for the loss of accuracy. We find that pruning 40% of filters achieves a good trade-off for the final architecture – reducing accuracy by less than 10% (< 0.5 px EPE) for more than 40% speed-up (Tab. 10). We re-train the pruned CNN with the same dataset schedule and settings as initial training, except for training on FlyingChairs [FDI*15] where we start with the lower learning rate of 1×10^{-5} and train for fewer iterations as training converges quickly, e.g., a maximum of 15 epochs (1.5 hours).

We evaluate different options of pruning as post-training optimization. As our initial training consists of multiple stages with different datasets, we first evaluate after which stage to prune and

Table 8: Ablation of different light flow-estimator configurations.

Variant	Sintel Final Train		Number of parameters		Desktop run-time		iPad Air run-time		iPad Pro run-time	
	EPE (↓)		M (↓)		FPS (↑)		FPS (↑)		FPS (↑)	
5 estimators, none separated	2.825		5.99		40.26		2.83		5.09	
5 estimators, all separated	3.813	+34.9%	2.66	-55.6%	43.49	+8.0%	4.52	+59.7%	7.76	+52.4%
5 estimators, last two separated	3.104	+9.8%	4.77	-20.3%	44.06	+9.4%	4.22	+49.1%	7.28	+43.0%
4 estimators, none separated	<u>3.229</u>	<u>+14.3%</u>	<u>5.31</u>	<u>-11.3%</u>	<u>79.12</u>	<u>+96.5%</u>	<u>6.17</u>	<u>+118.0%</u>	<u>10.41</u>	<u>+104.5%</u>
4 estimators, last separated	3.460	+22.4%	4.68	-21.8%	81.28	+101.7%	7.35	+159.7%	12.80	+151.4%

Table 9: Ablation of different refinement configurations.

Variant	Sintel Final Train		Number of parameters		Desktop run-time		iPad Air run-time		iPad Pro run-time	
	EPE (↓)		M (↓)		FPS (↑)		FPS (↑)		FPS (↑)	
4 estimators, default refinement	3.229		5.31		79.12		6.17		10.41	
4 estimators, no refinement	3.258	+0.9%	4.79	-9.8%	86.72	+9.6%	7.44	+20.5%	13.22	+27.0%
4 estimators, separated refinement	<u>3.169</u>	<u>-1.8%</u>	<u>4.85</u>	<u>-8.6%</u>	<u>83.53</u>	<u>+5.5%</u>	<u>7.23</u>	<u>+17.1%</u>	<u>12.87</u>	<u>+23.6%</u>
5 estimators, default refinement	2.825		5.99		40.26		2.83		5.06	
5 estimators, no refinement	3.248	+14.97%	5.47	-8.6%	51.80	+28.6%	4.22	+49.1%	7.31	+44.4%
5 estimators, separated refinement	2.922	+3.43%	5.53	-7.6%	47.43	+17.8%	3.47	+22.6%	6.00	+18.5%

Table 10: Ablation of different pruning amounts on desktop system.

Variant	Sintel Final Train		Number of parameters		Desktop run-time		iPad Air run-time		iPad Pro run-time	
	EPE (↓)		M (↓)		FPS (↑)		FPS (↑)		FPS (↑)	
4 estimators, separated refinement	3.169		4.85		79.12		7.23		12.87	
30% pruned	3.275	+3.3%	3.38	-30.3%	86.47	-9.3%	8.79	+21.5%	16.45	+27.8%
40% pruned	<u>3.443</u>	<u>+8.6%</u>	<u>2.83</u>	<u>-41.6%</u>	<u>86.61</u>	<u>-9.4%</u>	<u>10.00</u>	<u>+38.3%</u>	<u>18.94</u>	<u>+47.1%</u>
50% pruned	4.036	+27.3%	2.24	-53.8%	88.94	-12.4%	13.76	+90.3%	27.50	+113.6%
5 estimators	2.825		5.99		40.26		2.83		5.06	
30% pruned	2.900	+2.6%	4.06	-32.2%	46.79	-16.2%	3.89	+37.4%	6.95	+37.3%
40% pruned	3.022	+6.9%	3.35	-44.0%	50.68	-25.8%	4.63	+63.6%	8.17	+61.4%
50% pruned	3.106	+9.9%	3.01	-49.7%	51.51	-27.9%	4.86	+71.7%	8.47	+57.4%

Table 11: Ablation of different quantization options on mobile devices using our lite-flow CNN.

Quantization settings		CNN file size	Sintel Final Train		iPad Air run-time		iPad Pro run-time	
Number of bits	low-prec. acc.	Megabyte (MB) (↓)	EPE (↓)		FPS (↑)		FPS (↑)	
32-bit		16.3	3.577		9.95		18.94	
32-bit	✓	16.3	3.584	+0.2%	12.73	+27.9%	24.04	+26.9%
16-bit		8.2	3.577	±0.0%	10.00	+0.5%	18.64	-1.5%
16-bit	✓	8.2	3.584	+0.2%	12.68	+27.4%	23.96	+21.9%
8-bit		4.1	3.609	+0.9%	9.69	-2.6%	18.74	-1.0%
8-bit	✓	4.1	3.621	+1.2%	12.89	+29.5%	23.84	+27.2%
4-bit		2.1	6.665	+86.3%	9.77	-1.8%	18.75	-1.0%
4-bit	✓	2.1	6.665	+86.3%	12.93	+29.9%	23.81	+27.0%

Table 12: Ablation of pruning and re-training at different training stages. The Initial model is the one with 4 estimators and separated refinement and then we have its variations with respect to pruning at different stages. Note that increase in EPE is the least for the 3rd variant.

Variant	Training:	Chairs → Things3D → Sintel	
		EPE (↓) on Sintel Train	
Initial		3.169	
Prune 30% after Chairs, re-train from Chairs		3.505	+10.6%
Prune 30% after Things3D, re-train from Things3D		3.290	+3.8%
Prune 30% after Sintel, re-train from Chairs		3.275	+3.3%
Prune 30% after Sintel, re-train from Sintel		3.413	+7.7%

at which stage to start re-training. We observe that the best accuracy is obtained when pruning the fully trained CNN and re-training from the beginning of the dataset schedule (Tab. 12). We believe this works best as learned representations after only training on Chairs [FDI*15] or Things3D [MIH*16] are not as distinctive as after full training.

Next, we evaluate which trade-offs result from different amounts of pruned channels. For mobile devices and our final CNN, we find that pruning up to 40% of channels results in significant run-time improvement with plausible accuracy loss. Pruning 50% of channels results in a substantially higher accuracy loss (Tab. 10). For desktop, pruning – similar to reducing the number of channels (Tab. 7) – results in a smaller speed-up than on mobile devices. Considering only a small improvement of the already high frame rate in exchange for a significant loss in quality, we do not recommend pruning for the desktop version.

Quantization and Mobile Deployment: For mobile deployment, we make use of CoreML [App] as the framework for executing our CNNs on Apple mobile devices. We apply 8 bit linear weight-quantization and enable the accumulation of low-precision intermediate results (Tab. 11). This minimizes the file size by 75 % (compared to 32 bit weights) and further improves run-time performance by 30 % (on mobile devices) with only negligible accuracy loss. Further analysis shows that our method does not profit from using the on-device Neural Processing Unit (NPU).

1.2. Implementation Details

Pruning. For convolution filter pruning we use a PyTorch [P*19] implementation by Gongfan Fang [Fan19]. We use l_1 strategy for selecting filters to prune. l_2 strategy is available too but Li *et al.* [LKD*16] show that these strategies perform comparable. We round the number of resulting channels to a multiple of 8, as other filter counts result in a run-time overhead on mobile devices. We re-train the pruned CNN with the same dataset schedule and settings as initial training, except for training on FlyingChairs [FDI*15] where we start with the lower learning rate

of 1×10^{-5} and train for fewer iterations as training converges quickly, e.g., a maximum of 15 epochs (1.5 hours).

Mobile execution. We use CoreML [App] as the framework for executing our CNNs on Apple mobile devices. We evaluate using iPad Pro (11-inch, 2nd gen 2020) and iPad Air (3rd gen, 2019). CoreML efficiently implements standard CNN operations, however, the two operations specific to optical flow, i.e., correlation and warping, need to be implemented as custom layers using Metal GPU shaders for parallel and efficient computation using the mobile GPU. After CNN conversion to CoreML, we apply 8-bit linear weight quantization using coremltools, low precision accumulation is enabled to enforce low precision accumulation in all operations.

1.3. Training Settings

Similar to the original PWC-Net [SYLK18], we train our mobile architecture on the training dataset schedule FlyingChairs [FDI*15] → FlyingThings3D → Sintel [BWSB12]. Tab. 13 lists training settings for respective stages. We compute the multi-scale losses by taking the per-pixel difference between the output of each flow estimator and an accordingly downscaled ground-truth optical flow. The pixel-level loss values are summed up to a final single value which is then used as a training objective by the optimizer. Like PWC-Net [SYLK18], we scale the ground-truth optical flow with a factor of 20 prior to calculating the loss and thus have to divide the flow estimate by 20 at test time. For training we use AdamW optimizer [LH19] with $\beta_1 = 0.09$, $\beta_2 = 0.99$, and l_2 weight regularization with trade-off $\gamma = 0.0004$.

Given predicted flow field uv_{Pred} and ground-truth flow field uv_{GT} , EPE loss (Eqn. 10) and a robust l_1 loss (Eqn. 11) is defined as follows:

$$\mathcal{L}_{EPE}(uv_{Pred}, uv_{GT}) = \frac{1}{W \cdot H} \sum_{x,y} \|uv_{Pred}(x,y) - uv_{GT}(x,y)\|_2 \quad (10)$$

$$\mathcal{L}_{robust}(uv_{Pred}, uv_{GT}) = \frac{1}{W \cdot H} \sum_{x,y} (\|uv_{Pred}(x,y) - uv_{GT}(x,y)\|_1 + \epsilon)^q \quad (11)$$

with typical values of $\epsilon = 0.01$ and $q = 0.4$. $q < 1$ results in less penalty to large error values and thus makes the loss more robust to outliers, which is necessary for fine-tuning on realistic, difficult datasets [SYLK18].

1.4. Hyperparameters Configuration

We evaluate different hyperparameters while training the original PWC-Net [SYLK18] to determine a baseline that achieves the best possible accuracy. We find that AdamW optimizer [LH19] reaches a significantly better accuracy than Adam [KB15] – with and without l_2 weight regularization (Tab. 14). Furthermore, we find that equally weighted multi-scale losses – as opposed to commonly used exponential weighting [SYLK18, HTL18, HTL20, YR19b] –

Table 13: Our training settings for different datasets. Training times are specified for one Nvidia V100 GPU.

	FlyingChairs [FDI* 15]	FlyingThings3D [MIH* 16]	Sintel (Final) [BWSB12]
Batch size	8	4	4
Dataset resolution	512×384	960×540	1024×436
Training resolution	448×320	768×384	448×384
Loss function	EPE (Eqn. 10)	robust l_1 (Eqn. 11) $q = 0.4, \epsilon = 0.01$	robust l_1 (Eqn. 11) $q = 0.4, \epsilon = 0.01$
Initial learning rate	1×10^{-4}	1×10^{-5}	$5 \cdot 1 \times 10^{-5}$
Learning rate schedule	as S_{long} in [IMS*17]	as S_{fine} in [IMS*17]	as in [SYLK18] [†]
Total epochs	60	10	200
Total iterations	1200K	400K	200K
Training time	6 hours	10 hours	4 hours

Table 14: Ablation of different hyperparameters for baseline training on desktop. Sintel Final EPE is measured after training the original PWC-Net [SYLK18] architecture for 30 epochs on FlyingChairs [FDI* 15].

Hyperparameters				Sintel Final Train
Optimizer	Loss weights	Regularization	Gradient stopping	EPE (\downarrow)
Adam	exponential			5.561
Adam	exponential	l_2		6.938
AdamW	exponential			5.153
AdamW	exponential	l_2		5.263
<u>AdamW</u>	equal	l_2		<u>5.086</u>
AdamW	exponential	l_2	✓	6.269
AdamW	equal	l_2	✓	5.465

result in slightly better accuracy. Additionally, we consider gradient stopping as proposed by Hofinger *et al.* [HBP*20], but observe that it does not result in any improvement.

1.5. Data Augmentation

Dosovitskiy *et al.* [FDI* 15] found that augmentations are important for learning-based optical flow methods to prevent overfitting on synthetic training data and to ensure generalization for real-world data. Similarly, we apply geometric and color transformations to input frames and corresponding flow fields, as listed in tab. 15. Geometric transformations are applied equally to both frames of an input pair, and must be reflected accordingly in the flow field. For example, a translation applied to the frames requires a translation applied to the flow field; a rotation of frames requires a rotation of the flow field and its motion vectors. Color transformations need to be applied only to the frames, not the flow field. While it would be possible to apply different transformations (geometric, colors) per frame of an input pair [TD20] – to further increase robustness against illumination changes for example – we find that transformations applied equally per frame pair are sufficient augmentations that prevent overfitting, while not making training of small CNN variants too difficult [HZC* 17].

Table 15: Our augmentation settings. The same settings are applied to all training stages (tab. 13), except when fine-tuning on Sintel [BWSB12] where no Gaussian noise is added.

Augmentation	Value range	Probability
Rotation	uniform in $[-17^\circ; 17^\circ]$	0.2
Random crop	training resolution, cf. tab. 13	1.0
Horizontal flip		0.5
Vertical flip		0.5
Additive brightness	normal, $\sigma = 0.02$	1.0
RGB multiplier	uniform in $[0.9; 1.1]$	1.0
Contrast multiplier	uniform in $[0.7; 1.3]$	1.0
Gamma adjustment	uniform in $[0.7; 1.5]$	1.0
RGB random order		1.0
Additive gaussian noise	σ uniform from $[0; 0.04]$	1.0

2. Warping Error

In Tab. 16 we show the warping error (using ℓ_1 metric, as defined in the main paper) over the stylization tasks following Lai *et al.* [LHW* 18].

3. Extended User Study

To also compare with the methods of Shekhar *et al.* [SST* 19] and Thimonier *et al.* [TDKP21] we conducted another user study, involving only these two methods. The setup is similar to the one described in the main paper except that it was performed by a different group of participants to avoid bias. In total, 12 persons (3 female, 8 male, and 1 did not specify) between the ages of 25 to 40 years participated in the study. Fig. 15 shows that our method surpasses the other methods by a large margin.

4. More Optical-Flow Results

In Fig. 16 and Fig. 17 we show further results for our lite optical flow network (configured as presented in the main paper) compared to other methods on Sintel and DAVIS.

Task	Warping Error: L1							
	DAVIS				VIDEVO			
	V_p	[BTS*15]	[LHW*18]	Ours	V_p	[BTS*15]	[LHW*18]	Ours
CycleGAN/photo2ukiyoe	0.037	0.028	0.028	0.033	0.036	0.026	0.028	0.032
CycleGAN/photo2vangogh	0.049	0.033	0.037	0.042	0.047	0.032	0.036	0.041
fast-neural-style/rain-princess	0.079	0.043	0.055	0.062	0.076	0.040	0.055	0.061
fast-neural-style/udnie	0.044	0.025	0.031	0.036	0.038	0.021	0.026	0.031
WCT/antimonocromatismo	0.052	0.030	0.036	0.041	0.046	0.023	0.030	0.035
WCT/asheville	0.067	0.040	0.049	0.056	0.061	0.033	0.042	0.049
WCT/candy	0.065	0.035	0.045	0.051	0.058	0.029	0.038	0.045
WCT/feathers	0.058	0.035	0.040	0.049	0.054	0.030	0.037	0.045
WCT/sketch	0.054	0.037	0.038	0.046	0.050	0.032	0.035	0.041
WCT/wave	0.052	0.033	0.037	0.044	0.048	0.029	0.034	0.040
Average	0.056	0.034	0.040	0.046	0.051	0.036	0.036	0.042

Table 16: Flow Warping Error over stylization tasks. The optical flow evaluation is computed using GMA [JCL*21]

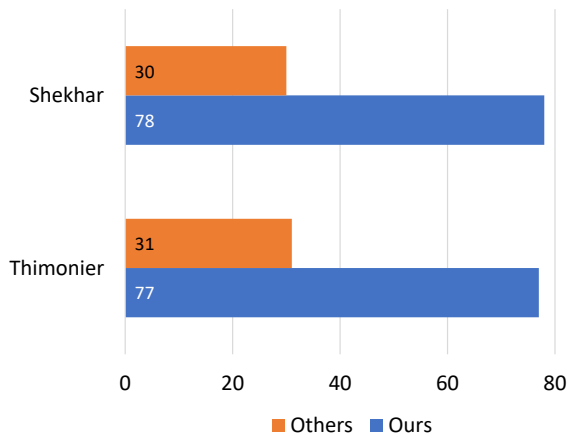


Figure 15: Statistics of the user study results on the removal of temporal flickering from per-frame stylized videos. For 12 participants and 9 different videos, we compare our method against Shekhar et al. and Thimonier et al. through a total of 108 randomized A/B tests.

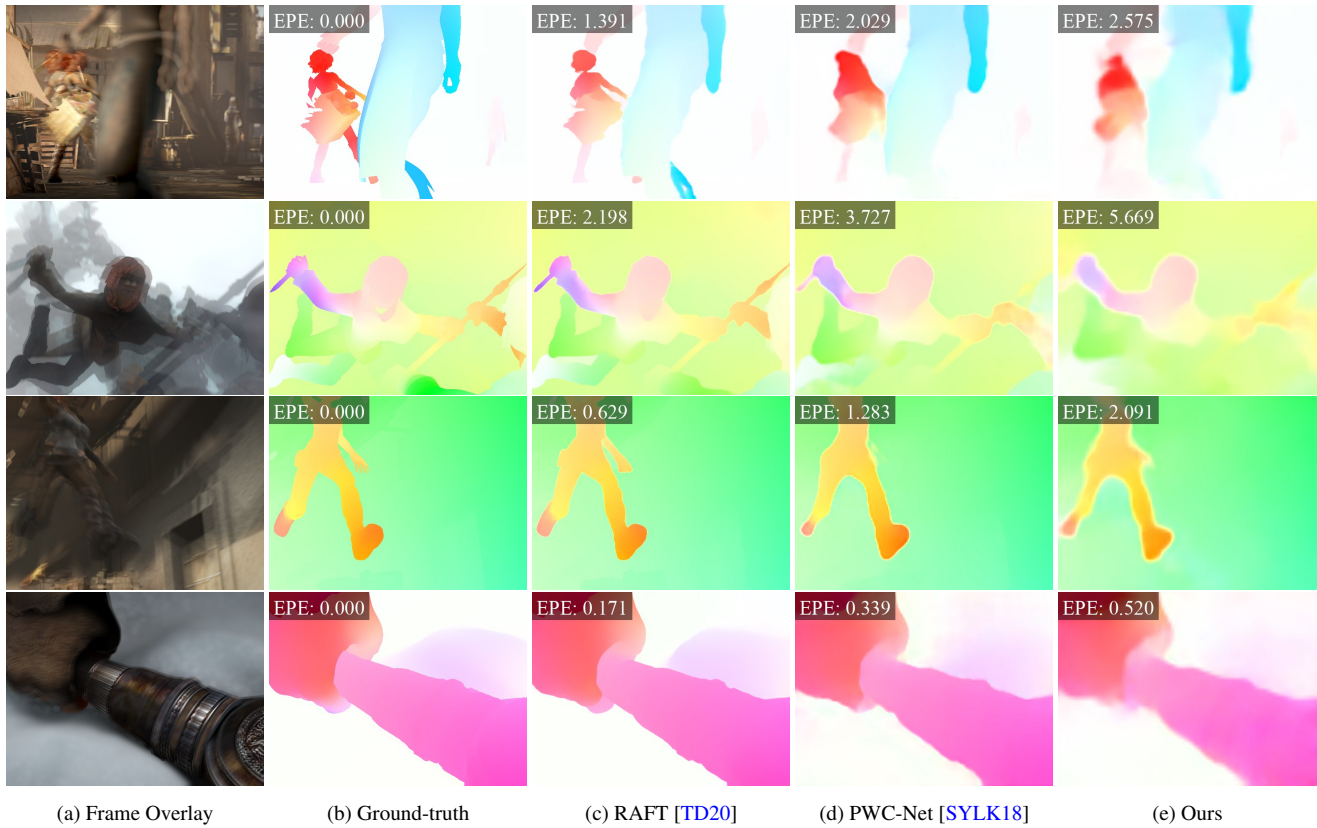


Figure 16: Optical flow estimated using the synthetic Sintel dataset [BWSB12].

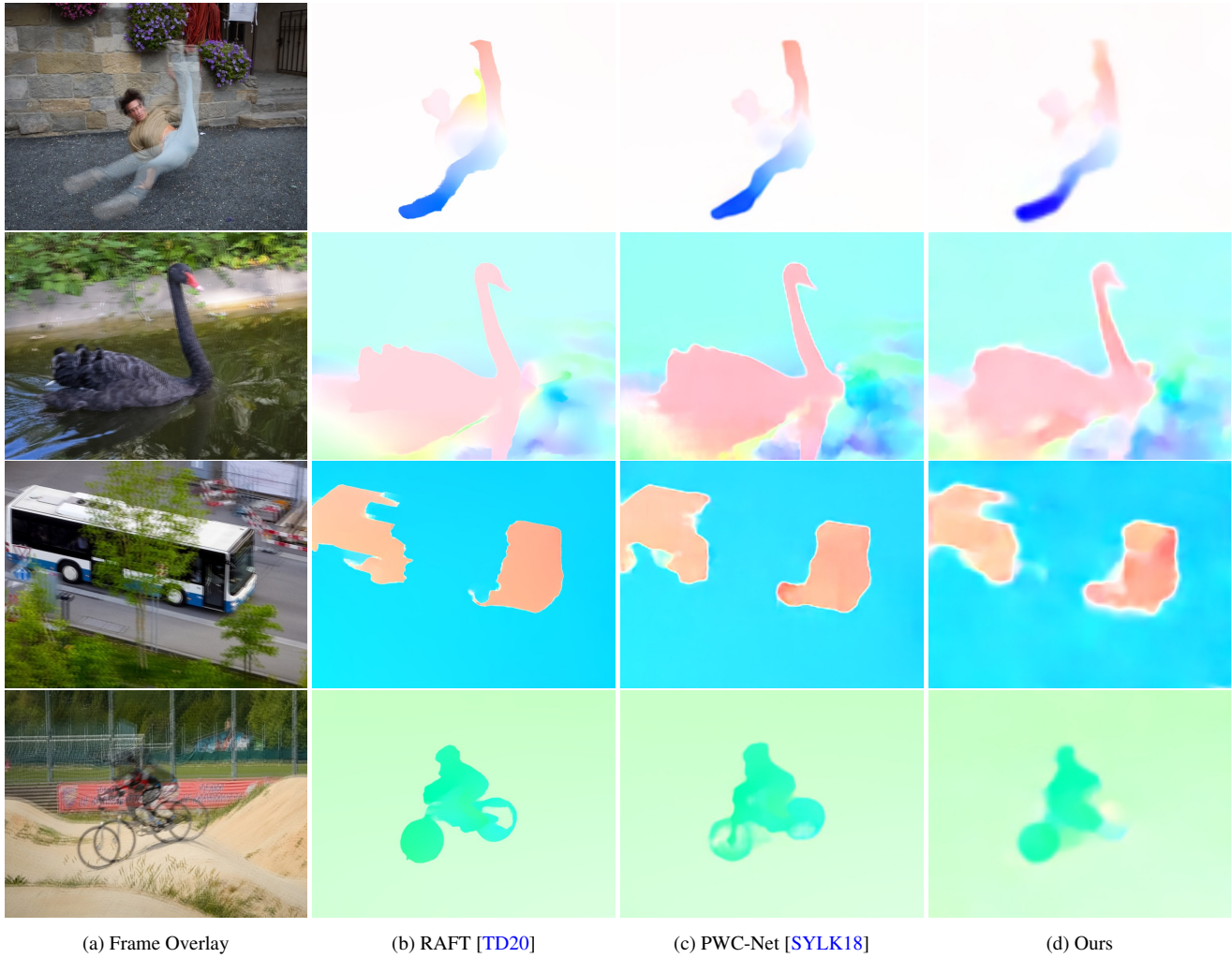


Figure 17: Optical flow estimated for the real-world dataset DAVIS [PTPC*17].