Interactive Video Stylization Using Few-Shot Patch-Based Training

ONDŘEJ TEXLER, DAVID FUTSCHIK, MICHAL KUČERA, ONDŘEJ JAMRIŠKA, and ŠÁRKA SOCHOROVÁ,
Czech Technical University in Prague, Faculty of Electrical Engineering
MENGLEI CHAI and SERGEY TULYAKOV, Snap Inc.
DANIEL SÝKORA, Czech Technical University in Prague, Faculty of Electrical Engineering

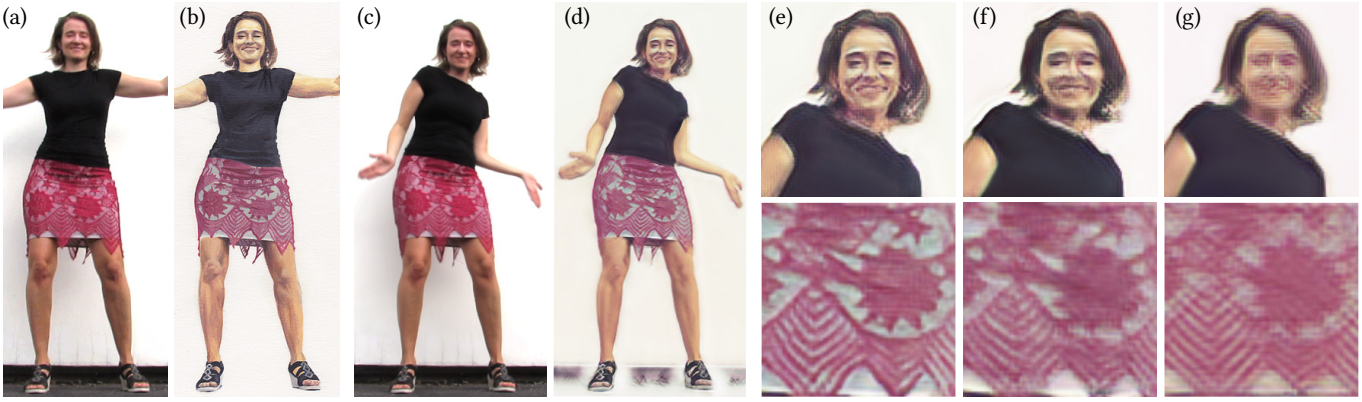

Fig. 1. An example of a sequence stylized using our approach. One frame from the original sequence is selected as a keyframe (a) and an artist stylizes it with acrylic paint (b). We use this single style exemplar as the only data to train a network. After 16 seconds of training, the network can stylize the entire sequence in real-time (c-d) while maintaining the state-of-the-art visual quality and temporal coherence. See the zoom-in views (e-g); even after 2 seconds of training, important structures already start to show up. Video frames (a, c) and style exemplar (b) courtesy of © Zuzana Studená.

In this paper, we present a learning-based method to the keyframe-based video stylization that allows an artist to propagate the style from a few selected keyframes to the rest of the sequence. Its key advantage is that the resulting stylization is semantically meaningful, i.e., specific parts of moving objects are stylized according to the artist's intention. In contrast to previous style transfer techniques, our approach does not require any lengthy pre-training process nor a large training dataset. We demonstrate how to train an appearance translation network from scratch using only a few stylized exemplars while implicitly preserving temporal consistency. This leads to a video stylization framework that supports real-time inference, parallel processing, and random access to an arbitrary output frame. It can also merge the content from multiple keyframes without the need to perform an explicit blending operation. We demonstrate its practical utility in various interactive scenarios, where the user paints over a selected keyframe and sees her style transferred to an existing recorded sequence or a live video stream.

Authors' addresses: Ondřej Texler, texleond@fel.cvut.cz; David Futschik, futscdav@fel.cvut.cz; Michal Kučera, kucerm22@fel.cvut.cz; Ondřej Jamriška, jamriond@fel.cvut.cz; Šárka Sochorová, sochosar@fel.cvut.cz, Czech Technical University in Prague, Faculty of Electrical Engineering, Karlovo náměstí 13, Praha 2, Czech Republic, 121 35; Menglei Chai, mchai@snap.com; Sergey Tulyakov, stulyakov@snap.com, Snap Inc., 2772 Donald Douglas Loop N, Santa Monica, United States, CA 90405; Daniel Sýkora, sykorad@fel.cvut.cz, Czech Technical University in Prague, Faculty of Electrical Engineering, Karlovo náměstí 13, Praha 2, Czech Republic, 121 35.

CCS Concepts: • **Computing methodologies** → **Motion processing**; **Image processing**.

Additional Key Words and Phrases: example-based, apperance translation, style transfer

ACM Reference Format:
Ondřej Texler, David Futschik, Michal Kučera, Ondřej Jamriška, Šárka Sochorová, Menglei Chai, Sergey Tulyakov, and Daniel Sýkora. 2020. Interactive Video Stylization Using Few-Shot Patch-Based Training. *ACM Trans. Graph.* 39, 4, Article 1 (July 2020), 11 pages. https://doi.org/10.1145/3386569.3392453

1 INTRODUCTION

Example-based stylization of videos became recently popular thanks to significant advances made in neural techniques [Kotovenko et al. 2019; Ruder et al. 2018; Sanakoyeu et al. 2018]. Those extend the seminal approach of Gatys et al. [2016] into the video domain and improve the quality by adding specific style-aware content losses. Although these techniques can deliver impressive stylization results on various exemplars, they still suffer from the key limitation of being difficult to control. This is due to the fact that they only measure statistical correlations and thus do not guarantee that specific parts of the video will be stylized according to the artist's intention, which is an essential requirement for use in a real production pipeline.

This important aspect is addressed by a concurrent approach—the keyframe-based video stylization [Bénard et al. 2013; Jamriška et al. 2019]. Those techniques employ guided patch-based synthesis [Fišer et al. 2016; Hertzmann et al. 2001] to perform a semantically meaningful transfer from a set of stylized keyframes to the rest of the

target video sequence. The great advantage of a guided scenario is that the user has a full control over the final appearance, as she can always refine the result by providing additional keyframes. Despite the clear benefits of this approach, there are still some challenges that need to be resolved to make the method suitable for a production environment.

One of the key limitations of keyframe-based stylization techniques is that they operate in a sequential fashion, i.e., their outputs are not *seekable*. When the user seeks to any given frame, all the preceding frames have to be processed first, before the desired result can be displayed. This sequential processing does not fit the mechanism of how frames are handled in professional video production tools, where random access and parallel processing are inevitable.

Another important aspect that needs to be addressed is merging, or blending, the stylized content from two or more (possibly inconsistent) keyframes to form the final sequence. Although various solutions exist to this problem (e.g., [Jamriška et al. 2019; Shechtman et al. 2010]), the resulting sequences usually suffer from visible clutter or ghosting artifacts. To prevent the issues with merging, the user has to resort to a tedious incremental workflow, where she starts by processing the whole sequence using only a single keyframe first. Next, she prepares a corrective keyframe by painting over the result of the previous synthesis run. This requires re-running the synthesis after each new correction, which leads to additional computational load and slows the overall process down.

To summarize, it would be highly beneficial to develop a guided style transfer algorithm that would act as a fast image filter. Such a filter would perform a semantically meaningful transfer on individual frames without the need to access past results, while still maintaining temporal coherence. In addition, it should also react adaptively to incoming user edits and seamlessly integrate them on the fly without having to perform an explicit merging.

Such a setting resembles the functionality of appearance translation networks [Isola et al. 2017; Wang et al. 2018a], which can give the desired look to a variety of images and videos. In these approaches, generalization is achieved by a large training dataset of aligned appearance exemplars. In our scenario, however, we only have one or a few stylized examples aligned with the input video frames, and we propagate the style to other frames with similar content. Although this may seem like a simpler task, we demonstrate that when existing appearance translation frameworks are applied to it naively, they lead to disturbing visual artifacts. Those are caused by their tendency to overfit the model when only a small set of appearance exemplars is available.

Our scenario is also similar to few-shot learning techniques [Liu et al. 2019; Wang et al. 2019b] where an initial model is trained first on a large generic dataset, and then in the inference time, additional appearance exemplars are provided to modify the target look. Although those methods deliver convincing results for a great variety of styles, they are limited only to specific target domains for which large generic training datasets exist (e.g., human bodies, faces, or street-view videos). Few-shot appearance translation to generic videos remains an open problem.

In this paper, we present a new appearance translation framework for arbitrary video sequences that can deliver semantically meaningful style transfer with temporal coherence without the need to perform any lengthy domain-specific pre-training. We introduce a patch-based training mechanism that significantly improves the ability of the image-to-image translation network to generalize in a setting where larger dataset of exemplars is not available. Using our approach, even after a couple of seconds of training, the network can stylize the entire sequence in parallel or a live video stream in real-time.

Our method unlocks a productive workflow, where the artist provides a stylized keyframe, and after a couple of seconds of training, she can watch the entire video stylized. Such rapid feedback allows the user to quickly provide localized changes and instantly see the impact on the stylized video. The artist can even participate in an interactive session and watch how the progress of her painting affects the target video in real-time. By replacing the target video with a live camera feed, our method enables an unprecedented scenario where the artist can stylize an actual live scene. When we point the camera at the artist's face, for instance, she can simultaneously paint the keyframe and watch a stylized video-portrait of herself. Those scenarios would be impossible to achieve with previous keyframe-based video stylization methods, and our framework thus opens the potential for new unconventional applications.

2 RELATED WORK

A straightforward approach to propagate the stylized content from a painted keyframe to the rest of the sequence could be to estimate dense correspondences between the painted keyframe and all other video frames [Li et al. 2019; Wang et al. 2019a] or compute an optical flow [Chen et al. 2013] between consecutive frames, and use it to propagate the stylized content from the keyframe. However, as shown in Jamriška et al. [2019] this simple approach may lead to noticeable distortion artifacts as the textural coherence is not maintained. Moreover, even when the distortion is small the texture advection effect leads to an unwanted perception that the stylized content is painted on the surface.

A more sophisticated approach to keyframe-based video stylization was pioneered by Bénard et al. [2013] who use guided patch-based synthesis [Hertzmann et al. 2001] to maintain textural coherence. In their approach a 3D renderer is used to produce a set of auxiliary channels, which guides the synthesis. This approach was recently extended to arbitrary videos by Jamriška et al. [2019]. In their framework, guiding channels are reconstructed automatically from the input video. Jamriška et al. also offer a post-processing step that merges the content stylized from multiple possibly inconsistent keyframes. Although patch-based techniques prove to deliver convincing results, their crucial drawback is that they can stylize the video only sequentially and require an explicit merging step to be performed when multiple keyframes are provided. Those limitations hinder random access, parallel processing, or real-time response, which we would like to preserve in our video stylization framework.

When considering fast video stylization, appearance translation networks [Isola et al. 2017] could provide a more appropriate solution. Once trained, they can perform semantically meaningful appearance transfer in real-time as recently demonstrated on human portraits [Futschik et al. 2019]. Nevertheless, a critical drawback here is that to learn such a translation network a large training

dataset is required. That can be hardly accessible in a generic video stylization scenario, where only a few hand-drawn exemplars exist, let alone in the context of video-to-video translation [Chan et al. 2019; Wang et al. 2018a] which is completely intractable.

Recently, few-shot learning techniques were introduced [Wang et al. 2019c,b] to perform appearance translation without the need to have a large dataset of specific style translation pairs. However, to do that a domain-specific dataset is required (e.g., facial videos, human bodies in motion, etc.) to pre-train the network. Such a requirement impedes the usage of previous few-shot methods in a general context where the target domain is not known beforehand.

In our method, we relax the requirement of domain-specific pre-training and show how to train the appearance translation network solely on exemplars provided by the user. Our approach bears resemblance to previous neural texture synthesis techniques [Li and Wand 2016; Ulyanov et al. 2016], which train a network with limited receptive field on a single exemplar image and then use it to infer larger textures that retain essential low-level characteristics of the exemplary image. A key idea here is to leverage the fully convolutional nature of the neural net. Even if the network is trained on a smaller patches it can be used to synthesize larger images.

Recently, the idea of patch-based training was further explored to accelerate training [Shocher et al. 2018] or to maintain high-level context [Shaham et al. 2019; Shocher et al. 2019; Zhou et al. 2018]; however, all those techniques deal only with a singe image scenario and are not directly applicable in our context. Also, they do not use a deliberately smaller batch of randomly cropped patches as a means of overfitting avoidance which is one of our key contributions.

Handling temporal consistency is a central task of video stylization methods. When individual frames are stylized independently, the resulting stylized animation usually contains intense temporal flickering. Although this effect is natural for traditional hand-colored animations [Fišer et al. 2014] it may become uncomfortable for the observer when watched for a longer period of time. Due to this reason, previous video stylization methods, either patch-based [Bénard et al. 2013; Fišer et al. 2017; Frigo et al. 2019; Jamriška et al. 2019] or neural-based [Chen et al. 2017; Ruder et al. 2018; Sanakoyeu et al. 2018], try to ensure temporal stability explicitly, e.g., by measuring the consistency between previous and a newly generated video frame. Alternatively, blind temporal coherency [Lai et al. 2018] could be used in the post-processing step. Yet, these approaches introduce data-dependency to the processing pipeline, which we would like to avoid to enable random access and parallel processing.

Our approach bears also a resemblance to a just-in-time training recently proposed by Mullapudi et al. [2019]. In their approach, labelling is provided for a subset of frames by a more accurate predictor and then propagated the the rest of the sequence using a quickly trained lightweight network. To deliver sufficient quality, a relatively large number of keyframes is necessary. Also, full-frame training is employed which we demonstrate could suffer from strong overfitting artifacts and thus is not applicable in our scenario where a detailed texture needs to be propagated.

3 OUR APPROACH

The input to our method is a video sequence I, which consists of N frames. Optionally, every frame I_i can be accompanied by a mask M_i to delineate the region of interest; otherwise, the entire video frame is stylized. Additionally, the user also specifies a set of keyframes $I^k \subset I$, and for each of them, the user provides stylized keyframes S^k, in which the original video content is stylized. The user can stylize the entire keyframe or only a selected subset of pixels. In the latter case, additional keyframe masks M^k are provided to determine the location of stylized regions (see Fig. 2 for details).

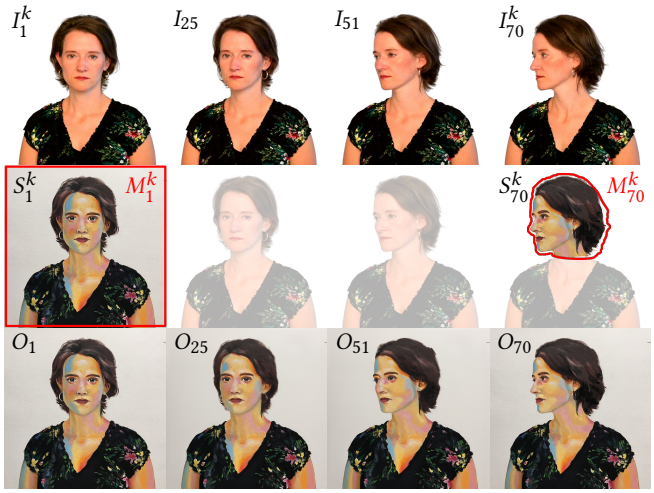

Fig. 2. The setting of video stylization with keyframes. The first row shows an input video sequence I. There are two keyframes painted by the user, one keyframe is painted fully (S_1^k) and the other is painted only partially (S_{70}^k). Mask M_1^k denotes that the entire keyframe is used; mask M_{70}^k specifies only the head region. Our task is to stylize all frames of the input sequence I while preserving the artistic style of the keyframes. The sequence O in the bottom row shows the result of our method. Video frames (I) and style exemplars (S) courtesy of © Zuzana Studená.

Our task is to stylize I in a way that the style from S^k is transferred to the whole of I in a semantically meaningful way, i.e., the stylization of particular objects in the scene remains consistent. We denote the output sequence by O. The aim is to achieve visual quality and temporal consistency comparable to the state-of-the-art in the keyframe-based video stylization [Jamriška et al. 2019]. However, in contrast to this previous work, we would like to stylize the video frames in random order, possibly in-parallel, or on-demand in real-time, without the need to wait for previous frames to be stylized or to perform explicit merging of stylized content from different keyframes. In other words, we aim to design a translation filter that can quickly learn the style from a few heterogeneously hand-drawn exemplars S^k and then stylize the entire sequence I in parallel, or any single frame on demand. It would also be beneficial if the learning phase was fast and incremental so that the stylization of individual video frames could start immediately, and the stylization quality would progressively improve over time.

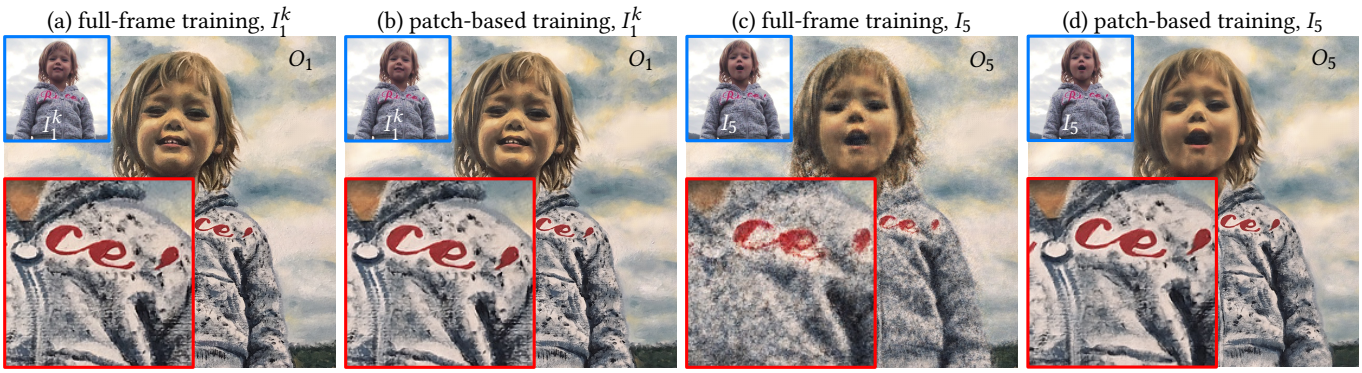

Fig. 3. Comparison of full-frame training vs. our patch-based approach: the original frames from the input sequence I are marked in blue and details of their stylized counterparts O are marked in red. The full-frame training scheme of Futschik et al. [2019] (a) as well as our patch-based approach (b) closely reproduce the frame on which the training was performed (see the frame S_1^k in Fig. 6). Both stylized frames (a, b) look nearly identical, although the training loss is lower for the full-frame scheme. Nevertheless, the situation changes dramatically when the two networks are used to stylize another frame from the same sequence (here frame I_5). The network which was trained using the full-frame scheme produces images that are very noisy and have fuzzy structure (c). This is due to the fact that the full-frame training causes the network to overfit the keyframe. The network is then unable to generalize to other frames in the sequence even though they structurally resemble the original keyframe. The network which was trained using our patch-based scheme retains the fidelity and preserves the important artistic details of the original style exemplar (d). This is thanks to the fact that our patch-based scheme better encourages the network to generalize to unseen video frames. Video frames (I) courtesy of © Zuzana Studená.

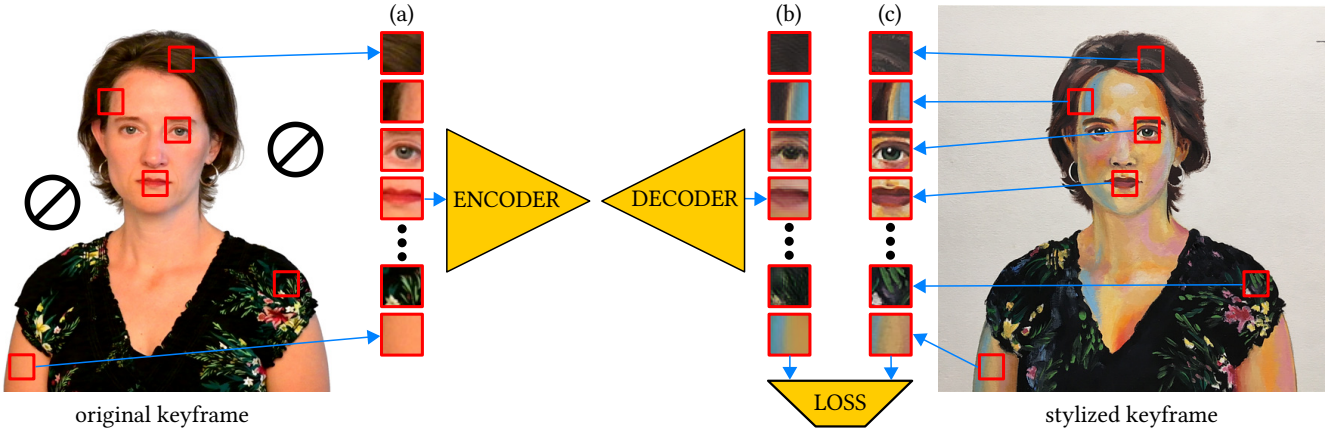

Fig. 4. Training strategy: we randomly sample a set of small patches from the masked area of the original keyframe (a). These patches are then propagated through the network in a single batch to produce their stylized counterparts (b). We then compute the loss of these stylized counterparts (b) with respect to the co-located patches sampled from the stylized keyframe (c) and back-propagate the error. Such a training scheme is not limited to any particular loss function; in this paper, we use a combination of L1 loss, adversarial loss, and VGG loss as described in [Futschik et al. 2019]. Video frame (left) and style exemplar (right) courtesy of © Zuzana Studená.

To design such a filter, we adopt the U-net-based image-to-image translation framework of Futschik et al. [2019], which was originally designed for the stylization of faces. It uses a custom network architecture that can retain important high-frequency details of the original style exemplar. Although their network can be applied in our scenario directly, the quality of results it produces is notably inferior as compared to current state-of-the-art (see Fig. 3c and our supplementary video at 2:20). One of the reasons why this happens is that the original Futschik et al.'s network is trained on a large dataset of style exemplars produced by FaceStyle algorithm [Fišer et al. 2017]. Such many exemplars are not available in our scenario, and thus the network suffers from strong overfitting. Due to this reason, keyframes can be perfectly reconstructed; however, the rest of the frames are stylized poorly, even after applying well-known data augmentation methods. See the detailed comparison in Figures 3 and 9. Furthermore, the resulting sequence also contains a disturbing amount of temporal flickering because the original method does not take into account temporal coherence explicitly.

To address the drawbacks mentioned above, we alter how the network is trained and formulate an optimization problem that allows fine-tuning the network's architecture and its hyper-parameters to get the stylization quality comparable to the current state-of-the-art,

even with only a few training exemplars available and within short training time. Also, we propose a solution to suppress temporal flicker without the need to measure consistency between individual video frames explicitly. In the following sections, those improvements are discussed in further detail.

3.1 Patch-Based Training Strategy

To avoid network overfitting to the few available keyframes, we adopt a patch-based training strategy. Instead of feeding the entire exemplar to the network as done in [Futschik et al. 2019], we randomly sample smaller rectangular patches from all stylized keyframes S^k (see Fig. 4) and train the network to predict a stylized rectangluar area of same size as input. The sampling is performed only within the area of masked pixels M^k. Note that thanks to the fully convolutional nature of the network, once trained, it can be directly used to stylize the entire video frame even though the training was performed on smaller patches (see Fig. 5). The key benefit of this explicit cropping and randomization step is that it simulates the scenario when a large and diverse dataset is used for training. It prevents the network from overfitting and generalizes to stylize the other video frames better. This training strategy is similar to one previously used for texture synthesis [Zhou et al. 2018].

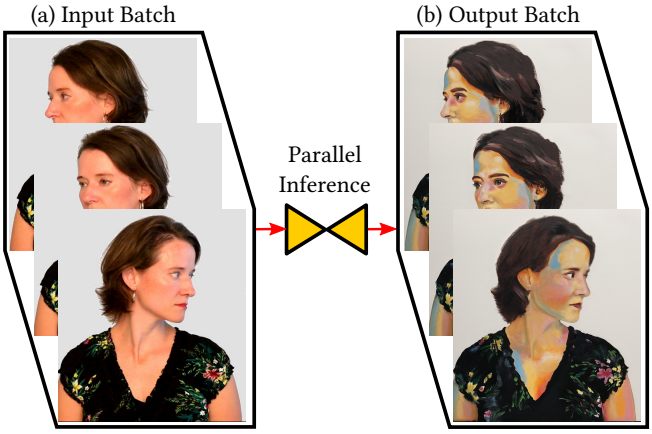

Fig. 5. Inference: thanks to the fully convolutional nature of the network, we can perform the inference on entire video frames, even though the training is done on small patches only. Since the inference does not depend on other stylized frames, all video frames can be stylized in parallel or in random order. This allows us to pass many or even all of the input frames (a) through the network in a single batch and get all output frames (b) at once. Video frames (left) courtesy of © Zuzana Studená.

Although the reconstruction loss measured on keyframes S^k is higher when compared to full-frame training after comparable amount of time, on the remaining frames of I the reconstruction loss is considerably lower when comparing to the frames stylized using state-of-the-art keyframe-based video stylization method of Jamriška et al. which we purposefully consider as a ground truth (cf. supplementary video at 0:08 and 1:08). This lower loss w.r.t. Jamriška et al. translates to much better visual quality.

3.2 Hyper-parameter Optimization

Although the patch-based training strategy considerably helps to resolve the overfitting problem, we find that it is still essential to have a proper setting of critical network hyper-parameters, as their naive values could lead to poor inference quality, especially when the training performance is of great importance in our applications (see Fig. 8). Besides that, we also need to balance the model size to capture the essential characteristics of the style yet being able to perform the inference in real-time using off-the-shelf graphics card.

We formulate an optimization problem in which we search for an optimal setting of the following hyper-parameters: W_p—size of a training patch, N_b—number of patches used in one training batch, α—learning rate, and N_r—number of ResNet blocks used in our network architecture. The aim is to minimize the loss function used in Futschik et al. [2019] computed over the frames inferred by our network and their counterparts stylized using the method of Jamriška et al. [2019]. The minimization is performed subject to the following hard constraints: T_t—the time for which we allow the network to be trained for and T_i—the inference time for a single video frame. Since T_t as well as T_i are relatively short (in our setting $T_t = 30$ and $T_i = 0.06$ seconds) full optimization of hyper-parameters becomes tractable. We used the grid search method on a GPU cluster, to find the optimal values (see detailed scheme Fig. 6). In-depth elaboration can be found in Section 4.

In our experiments, we found that hyper-parameter optimization is relatively consistent when different validation sequences are used. We thus believe the setting we found is useful for a greater variety of styles and sequences. Note also that the result of Jamriška et al. is used only for fine-tuning of hyper-parameters. Once this step is finished, our framework does not require any guided patch-based synthesis algorithm and can act fully independently.

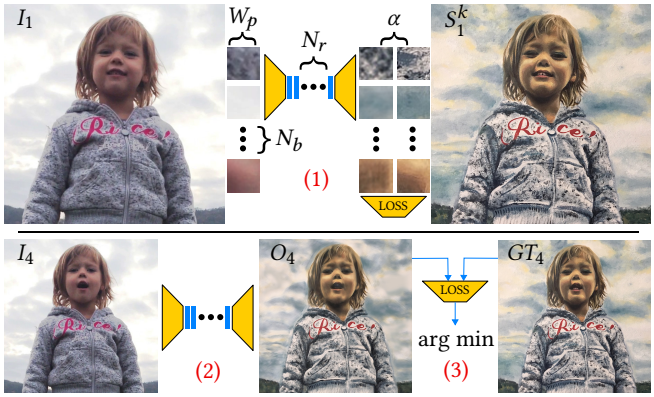

Fig. 6. To fine-tune critical hyper-parameters of our network, we propose the following optimization scheme. We tune batch size N_b, patch size W_p, number of ResNet blocks N_r, and learning rate α. Using the grid search method we sample 4-dimensional space given by these hyper-parameters and for every hyper-parameter setting we (1) perform a training for a given amount of time, (2) do inference on unseen frames, and (3) compute the loss between inferred frames (O_4) and result of [Jamriška et al. 2019] (GT_4) - which we consider to be ground truth. The objective is to minimize this loss. Note that the loss in step (1) and the loss in step (3) are both the same. Video frames (I) and style exemplar (S) courtesy of © Zuzana Studená.

3.3 Temporal Coherency

Once the translation network with optimized hyper-parameters is trained using the proposed patch-based scheme, style transfer to I can be performed in real-time or in parallel on the off-the-shelf graphics card. Even though such a frame-independent process yields relatively good temporal coherence on its own (as noted by Futschik et al.), in many cases, temporal flicker is still apparent. We aim to suppress it while keeping the ability of the network to perform frame-independent inference. We analyzed the source of the temporal instability and found two main reasons: (1) temporal noise in the original video and (2) visual ambiguity of the stylized content. We discuss our solution to those issues in the following paragraphs.

We observed that the appearance translation network tends to amplify temporal noise in the input video, i.e., even a small amount of temporal instability in the input video causes visible flicker in the output sequence. To suppress it, we use the motion-compensated variant of bilateral filter operating in the temporal domain [Bennett and McMillan 2005]. See our supplementary video (at 2:40) for the flicker reduction that can be achieved using this pre-filtering. Although bilateral filter requires nearby frames to be fetched into the memory, it does not violate our requirement for frame-independent processing.

Another observation we made is that filtering the input video reduces temporal flicker only on objects that have distinct and variable texture. Those that lack sufficient discriminatory information (e.g., homogeneous regions) flicker due to the fact that the visual ambiguity correlates with the network's ability to recall the desired appearance. To suppress this phenomenon, one possibility is to prepare the scene to contain only well distinctive regions. However, such an adjustment may not always be feasible in practice.

Instead, we provide an additional input layer to the network that will improve its discriminative power explicitly. This layer consists of a sparse set of randomly distributed 2D Gaussians, each of which has a distinct randomly generated color. Their mixture represents a unique color variation that helps the network to identify local context and suppress the ambiguity (see Fig. 7). To compensate for the motion in the input video, Gaussians are treated as points attached to a grid, which is deformed using as-rigid-as-possible (ARAP) image registration technique [Sýkora et al. 2009]. In this approach, two steps are iterated: (1) block-matching estimates optimal translation of each point on the grid, and (2) rigidity is locally enforced using the ARAP deformation model to regularize the grid structure. As this registration scheme can be applied independently for each video frame, the condition on frame independence is still satisfied.

The reason why the mixture of Gaussians is used instead of directly encoding pixel coordinates as done, e.g., in [Jamriška et al. 2019; Liu et al. 2018] is the fact that random colorization provides better localization and their sparsity, together with rotational symmetry, reduces the effect of local distortion, which may confuse the network. In our supplementary video (at 3:20) we, demonstrate the benefit of using the mixture of Gaussians over the layer with color-coded pixel coordinates. In case of extreme non-planar deformation (e.g., head rotation) or strong occlusion (multiple scene planes), additional keyframes need to be provided or the scene separated into multiple layers. Each keyframe or a scene layer has then its own dedicated deformation grid. We demonstrate this scenario in our supplementary video (at 2:56).

4 RESULTS

We implemented our approach in C++ and Python with PyTorch, adopting the structure of the appearance translation network of Futschik et al. [2019] and used their recommended settings including training loss. Ground truth stylized sequences for hyper-parameter tuning and comparison were produced using the video stylization method of Jamriška et al. [2019].

We performed fine-tuning of hyper-parameters on a selection of frames from our evaluation sequences. We computed their stylized counterparts using the method of Jamriška et al. [2019] and performed optimization using grid search on a cluster with 48 Nvidia Tesla V100 GPUs in 3 days. We searched over the following intervals: $W_p \in (12, 188)$, $N_b \in (5, 1000)$, $N_r \in (1, 40)$, $\alpha \in (0.0002, 0.0032)$. In total we sampled around 200,000 different settings of those hyper-parameters. We found the optimal patch size to be $W_p = 36$ pixels, the number of patches in one batch $N_b = 40$, learning rate $\alpha = 0.0004$, and the number of ResNet blocks $N_r = 7$.

See Fig. 8 to compare visual quality for different hyper-parameter settings. Note the substantial improvement in visual quality over different settings, which confirms the necessity of this optimization. An interesting outcome of the proposed hyper-parameter optimization is a relatively small number of patches in one batch $N_b = 40$ (Fig. 8a). This value interplays with our choice of patch-based training scheme. Although a common strategy would be to enlarge N_b as much as possible to utilize GPU capability, in our case, increasing N_b is actually counterproductive as it turns training scheme into a full-frame scenario that tends to overfit the network on the keyframe and produce poor results on unseen video frames. A smaller number of randomly selected patches in every batch increases the variety of back-propagation gradients and thus encourages the network to generalize better. From the optimal patch size $W_p = 36$ (Fig. 8b) it is apparent that smaller patches may not provide sufficient context, while larger patches may make the network less resistant to appearance changes caused by deformation of the target object and less sensitive to details. Surprisingly, the number of ResNet blocks N_r (see Fig. 8c) does not have a significant impact on the quality, although there is a subtle saddle point visible. Similar behavior also holds true for the learning rate parameter α. In addition, we also examined the influence of the number of network filters on the final visual quality (see our supplementary material). The measurements confirmed that the number of filters needs to be balanced as well to capture the stylized content while still avoid overfitting.

With all optimized hyper-parameters, a video sequence of resolution 640×640 with 10% of active pixels (inside the mask M^k) can be stylized in good quality at 17 frames per second after 16 seconds of training (see Fig. 1).

We evaluated our approach on a set of video sequences with different resolutions ranging from 350×350 to 960×540, containing different visual content (faces, human bodies, animals), and various

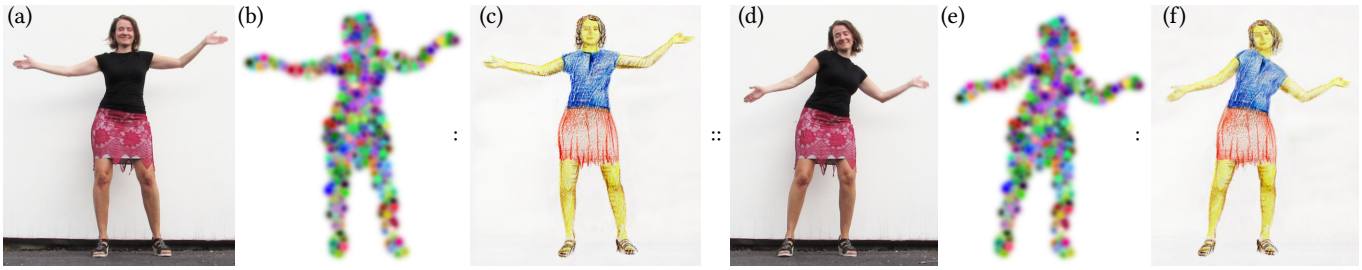

Fig. 7. To suppress visual ambiguity of the dark mostly homogeneous T-shirt in (a) an auxiliary input layer is provided that contains a mixture of randomly distributed and colored Gaussians (b). The translation network is trained on patches of which input pixels contain those additional color components. The aim is to reproduce the stylized counterpart (c). Once the network is trained a different frame from the sequence can be stylized (d) using adopted version of the auxiliary input layer (e). The resulting sequence of stylized frames (f) has notably better temporal stability (cf. our supplementary video at 2:40). Video frames (a, d) courtesy of © Zuzana Studená and style exemplar (b) courtesy of © Pavla Sýkorová.

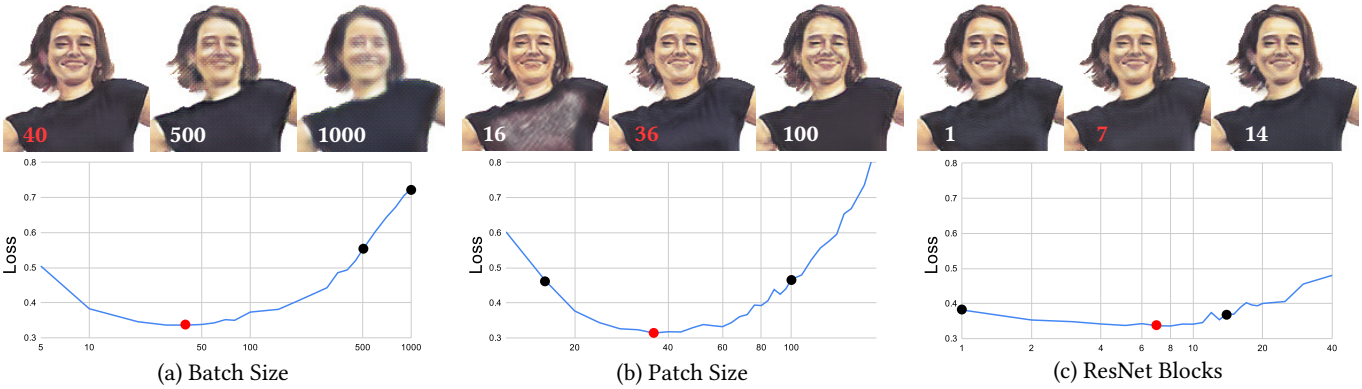

Fig. 8. Influence of important hyper-parameters on visual quality of results. The loss, y-axes, is computed w.r.t. the output of Jamriška et al. [2019]. The best setting for each hyper-parameter is highlighted in red: (a) The loss curve for the batch size N_b—the number of patches in one training batch (other hyper-parameters are fixed). As can be seen, increasing N_b deteriorates visual quality significantly; it indicates that there exists an ideal amount of data to pass through the network during the back-propagation step. (b) The loss curve for the patch size W_p. The optimal size of a patch is around 36x36 pixels. This fact indicates that smaller patches may not provide sufficient context while larger ones could make the network less robust to deformation changes. (c) The loss curve for the number of ResNet blocks N_r that corresponds to the capacity of the network. As can be seen, settings with 7 ResNet blocks is slightly better than other results; however, this hyper-parameter does have major impact on the quality of results. For additional experiments with hyper-parameter setting, refer to our supplementary text.

artistic styles (oil paint, acrylic paint, chalk, color pencil, markers, and digital image). Simpler sequences were stylized using only one keyframe (see Figures 1, 3, 7, 11, and 12) while the more complex ones have multiple (ranging from two to seven, see Figures 14, 13, 15, and 16). Before training, the target sequence was pre-filtered using the bilateral temporal filter. In case that the sequence contains regions having ambiguous appearances, we compute an auxiliary input layer with the mixture of randomly colored Gaussians that follows the motion in the target sequence. During the training phase, we randomly sample patches inside the mask M^k from all keyframes k and feed them in batches to the network to compute the loss and backpropagate the error. Training, as well as inference, were performed on Nvidia RTX 2080 GPU. The training time was set to be proportional to the number of input patches (number of pixels inside the mask M^k), e.g., 5 minutes for a 512×512 keyframe with all pixels inside the mask. After training, the entire sequence can be stylized at the speed of roughly 17 frames per second. See our supplementary video (at 0:08 and 1:08) for the resulting stylized sequences.

4.1 Comparison

To confirm the importance of our patch-based training strategy, we conducted comparisons with other commonly used methods for data-augmentation that can help avoiding overfitting such as adding Gaussian noise to the input, randomly erasing selected pixels, occluding larger parts of the input image, or performing dropout before each convolution layer. We found that none of these techniques can achieve comparable visual quality to our patch-based training strategy (see Fig. 9).

We compared our approach with the current state-of-the-art in keyframe-based video stylization [Jamriška et al. 2019]. For the results see Figures 10, 12, 14, 15, and our supplementary video (at 0:08 and 1:08). Note how the overall visual quality, as well as the temporal coherence, is comparable. In most cases, our approach

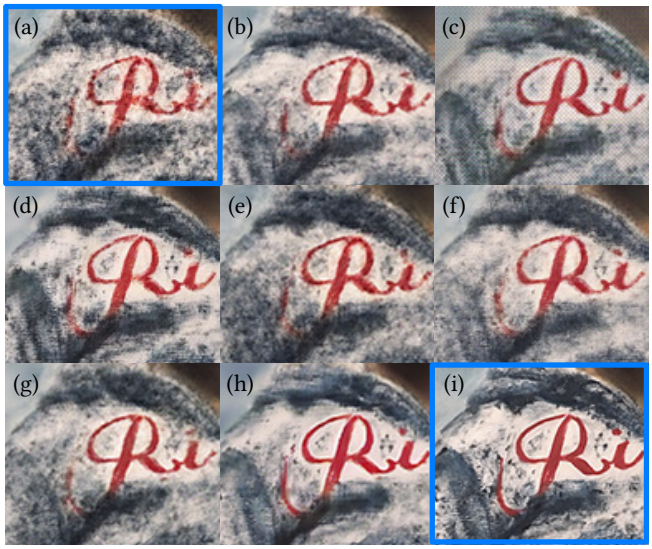

Fig. 9. To deal with the overfitting caused by a minimal amount of training data, we tried several commonly used techniques to enforce regularization. In all cases shown in this figure, we trained the network on the first frame; the shown results are zoomed details of the fifth frame. (a) is a result of the original full-frame training. (b-h) are results of full-frame training with some data augmentation. (i) is a result of our patch-based training strategy—see how our technique can deliver much sharper and significantly better visual quality results, please, zoom into the figure to better appreciate the difference. In case of (b-c), Gaussian noise was used to augment the data; (d) some pixels were randomly set to black; (e-f) some parts of the image were occluded; (g) dropout of entire 2D feature maps; (h) dropout of individual pixels before each convolution layer.

is better at preserving important structural details in the target video, whereas the method of Jamriška et al. often more faithfully preserves the texture of the original style exemplar. This is caused by the fact that the method of Jamriška et al. is non-parametric, i.e., it can copy larger chunks of the style bitmap to the target frame. Our method is parametric, and thus it can adapt to fine structural details in the target frame, which would otherwise be difficult to reproduce using bitmap chunks from the original style exemplar.

Regarding the temporal consistency, when our full-fledged flicker compensation based on the mixture of Gaussians is used our approach achieves comparable coherency in time to the method of Jamriška et al. It is also apparent that when multiple keyframes are used for stylization, ghosting artifacts mostly vanish in our method, unlike in Jamriška et al. When the original noisy sequence is used, or only the bilateral filtering is applied, the resulting sequence may flicker a little more when compared to the output of Jamriška et al. However, we argue that the benefits gained from random access and parallel processing greatly outweigh the slight increase of temporal flicker. Moreover, the order-independent processing brings also a qualitative improvement over the method of Jamriška et al. that tends to accumulate small errors during the course of the sequence, and visibly deteriorates after a certain number of frames.

Performance-wise a key benefit of our approach is that once the network is trained, one can perform stylization of a live video stream

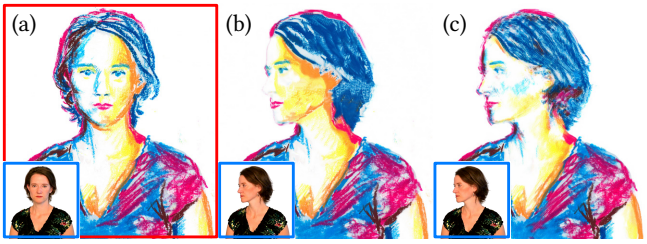

Fig. 10. When the target subject undergoes a substantial appearance change, the results of both Jamriška et al. [2019] (b) and our method (c) exhibit noticeable artifacts. The parts that were not present in the keyframe are reconstructed poorly—see the face and hair regions where [Jamriška et al. 2019] produces large flat areas, while our approach does not reproduce the color of the face well. Video frames (insets of a–c) and style exemplars (a) courtesy of © Zuzana Studená.

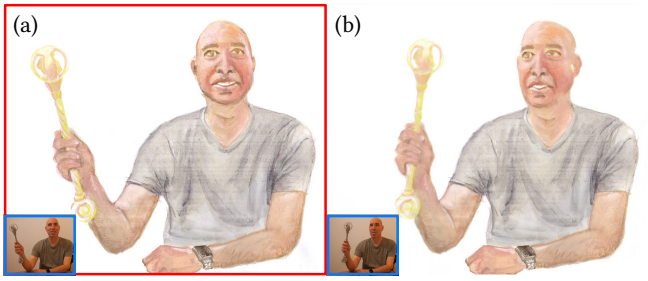

Fig. 11. Given one keyframe (a) and a video sequence (in blue), our method produces the stylized result (b). Video frames (insets of a, b) courtesy of © Adam Finkelstein and style exemplars (a) courtesy of © Pavla Sýkorová.

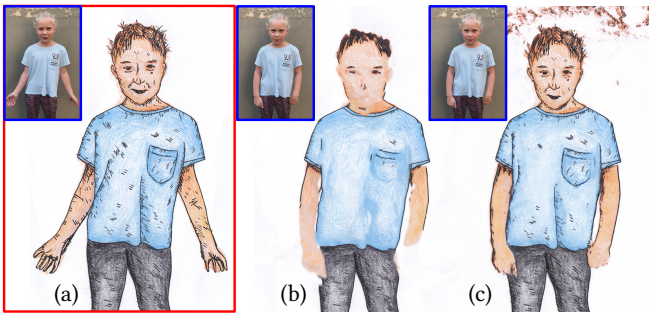

Fig. 12. For the state-of-the-art algorithm of [Jamriška et al. 2019], contour based styles (a) present a particular challenge (b). Using our approach (c), the contours are transferred with finer detail and remain sharp even as the sequence undergoes transformations. Video frames (insets of a–c) and style exemplar (a) courtesy of © Štěpánka Sýkorová.

in real-time. Even in the offline setting, when the training phase is taken into account, the overall end-to-end computation overhead is still competitive. On a 3 GHz quad-core CPU with Nvidia RTX 2080 GPU, a 512×512 sequence with 100 frames takes around 5 minutes to train until convergence and stylize using our approach, whereas the method of Jamriška et al. requires around 15 minutes.

Fig. 13. The Lynx sequence stylized using two keyframes (a, d). Notice how our method produces seamless transition between the keyframes while preserving fine texture of the style (b, c). Watch our supplementary video (at 1:22) to see the sequence in motion. Style exemplars (a, d) courtesy of © Jakub Javora.

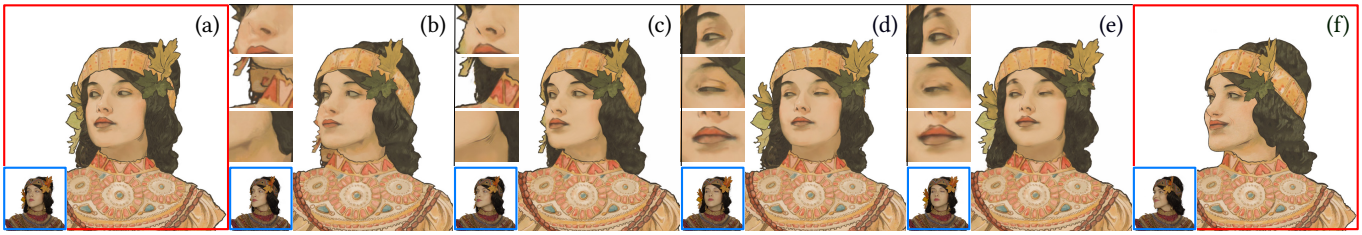

Fig. 14. Keyframes (a, f) were used to stylize the sequence of 154 frames. See the qualitative difference between Jamriška et al. [2019] (b) and our result (c). Focusing mainly on zoom-in views, our approach better preserves contour lines around the nose and chin; moreover, the method of Jamriška et al. suffers from blending artifacts—the face is blended into the hair region. On the other hand, comparison on a different frame from the same sequence shows that the result of Jamriška et al. (d) is qualitatively superior to our result (e) on this particular frame. See the corresponding zoom-in views where the approach of Jamriška et al. produces cleaner results. Video frames (insets of a–f) and style exemplars (a, f) courtesy of © Muchalogy.

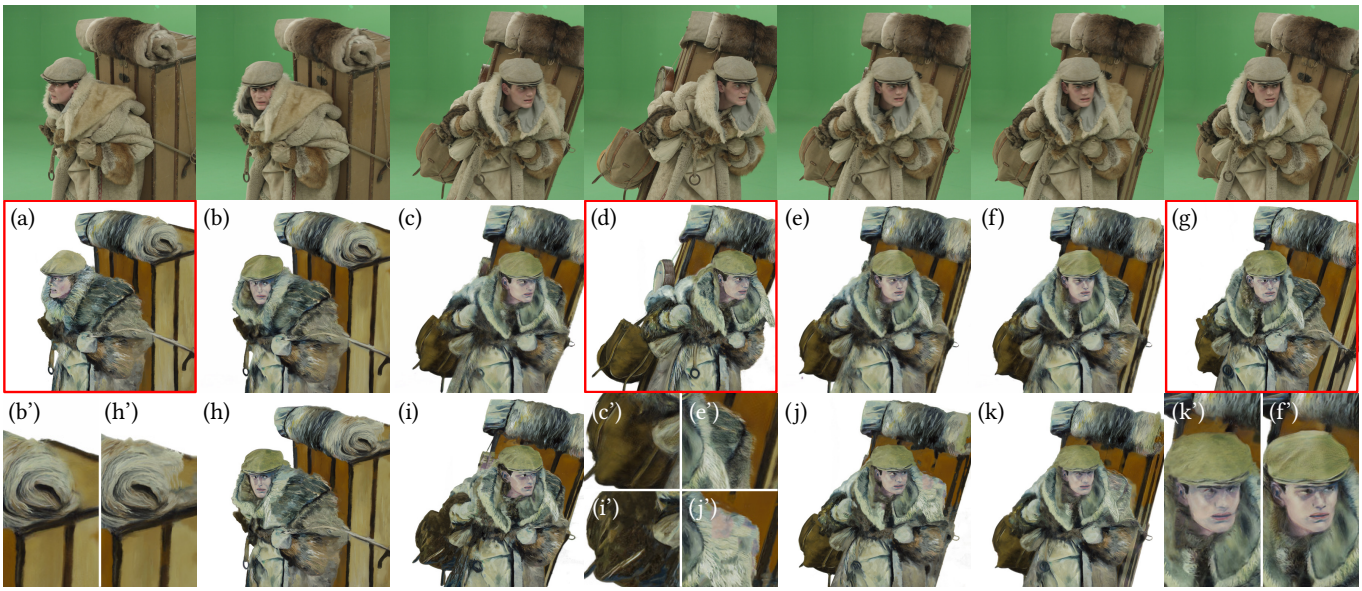

Fig. 15. A complex input sequence (the first row) with seven keyframes, three of them are shown in (a, d, g). Here we compare our approach to the approach of Jamriška et al. [2019]. See our result (b) and theirs (h) along with the close-ups (b', h'); due to their explicit handling of temporal coherence, the texture of the fur leaks into the box (h'). Next, compare our result (c) to theirs (i); our approach better reconstructs the bag (c', i'). Their issue with texture leakage manifests itself again on the shoulder in (j, j'), notice how our approach (e, e') produces a clean result. Lastly, see how our result (f, f') is sharper and the face is better pronounced compared to the result of Jamriška et al. [2019] (k, k'), which suffers from artifacts caused by their explicit merging of keyframes. Video frames (top row) and style exemplars (a, d, g) courtesy of © MAUR film.

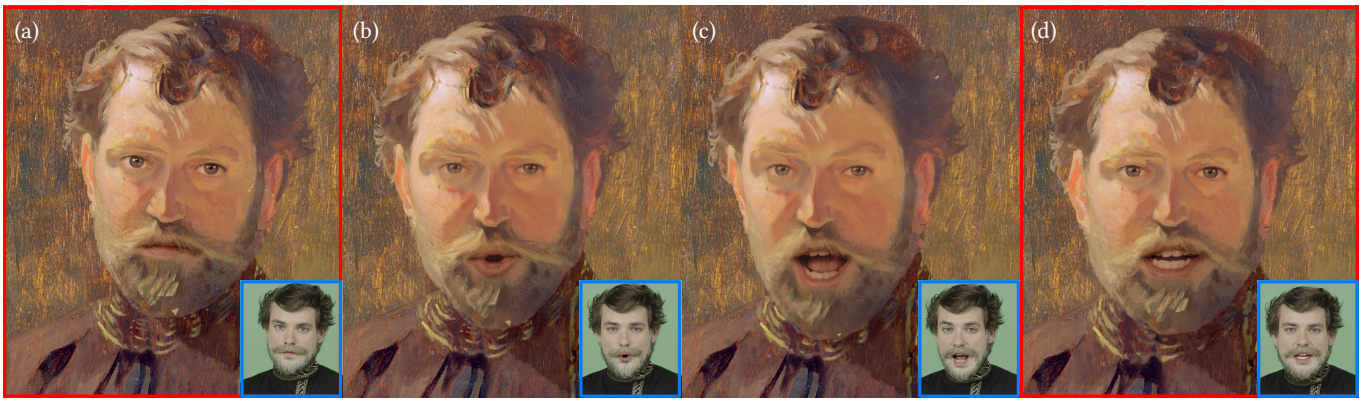

Fig. 16. An example sequence of 228 video frames (in blue) as stylized from two keyframes (a, d). Results of our method (b, c) stay true to style exemplars over the course of the sequence. Video frames (insets of a–d) and style exemplars (a, d) courtesy of © Muchalogy.

4.2 Interactive applications

To evaluate the ideas we presented in practice, we invited artists to work with our framework. We implement and experiment with three different setups in which the artists created physical as well as digital drawings. The goal of these sessions was to stylize one or more video keyframes artistically. Using a workstation PC, we provided the artists with a version of our framework that implements real-time interactive stylization of pre-prepared video sequences and stylization of live camera feeds.

These applications, all of which rely on and strongly benefit from the near real-time nature of patch-based training as well as the real-time performance of full-frame inference, naturally lend themselves to fast iteration. The artist is provided with real-time feedback that approximates what the final result of video stylization might look like, thus reducing the possibility of running into issues with artifacts that would be difficult to alleviate later on.

During the sessions, artists especially appreciated seeing video results very quickly, as it helps steer creative flow and offers the possibility of perceiving the effect of individual changes in the style exemplar at a glance. The overall experience was described as incredibly fun and paradigm-changing, with little to no negative feedback. Using this system is intuitive and even suitable for children. These different scenarios are described in detail in the supplementary material.

5 LIMITATIONS AND FUTURE WORK

Although our framework brings substantial improvements over the state-of-the-art and makes keyframe video stylization more flexible and interactive, there are still some limitations that could represent a potential for further research.

Despite the fact our technique uses different computational machinery than current state-of-the-art [Jamriška et al. 2019] (deep convolutional network vs. guided patch-based synthesis), both approaches share similar difficulties when stylized objects change their appearance substantially over time, e.g., when the object rotates and thus reveals some unseen content. Although our approach often resists slightly longer than patch-based synthesis due to the ability to generalize better, it usually cannot invent consistent stylization for new features that were not stylized in the original keyframe, see Fig. 10. In this case, the user needs to provide additional keyframes to make the stylization consistent.

As compared to the method of Jamriška et al. our approach may encounter difficulties when processing keyframes at a higher resolution (e.g., 4K) to stylize high-definition videos. Although the size of patches, as well as the network capacity, can be increased accordingly, the training may take notably longer time, as a different multi-scale approach [Wang et al. 2018b] could be necessary. However, the problem of training of larger models is an active research topic in machine learning, so we believe that soon, more efficient methods will be developed so that our technique would be applicable also at higher resolutions.

Although our approach does not require the presence of previous stylized frames to preserve temporal coherency, the motion-compensated bilateral filter, as well as the creation of layer with a random mixture of colored Gaussians, requires fetching multiple video frames. Even though those auxiliary calculations can still be performed in parallel, they need additional computation resources. Those may cause difficulties when considering real-time inference from live video streams. In our prototype, during the live capture sessions, treatment for improving temporal coherence was not taken into account. A fruitful avenue for future work would be to implement real-time variants of the motion-compensated bilateral filter as well as a mixture of colored Gaussians. Also, different methods could be developed that would enable the network to keep stylized video temporally coherent without the need to look into other video frames.

6 CONCLUSION

We presented a neural approach to keyframe-based stylization of arbitrary videos. With our technique, one can stylize the target sequence using only one or a few hand-drawn keyframes. In contrast to previous neural-based methods, our method does not require large domain-specific datasets nor lengthy pre-training. Thanks to our patch-based training scheme, optimized hyper-parameters,

and handling of temporal coherence, a standard appearance translation network can be trained on a small set of exemplars. Once trained, it can quickly deliver temporally coherent stylized videos with a visual quality comparable to the current state-of-the-art in keyframe-based video stylization, which uses guided patch-based synthesis. A key benefit of our technique is that it can work in a frame-independent mode, which is highly beneficial for current professional video editing tools that rely heavily on random access and parallel processing. It also does not require the explicit merging of stylized content when slightly inconsistent keyframes are used.

Moreover, since the network in our framework can be trained progressively, and the inference runs in real-time on off-the-shelf GPUs, we can propose several new video editing scenarios that were previously difficult to achieve. Those include stylization of a live video stream using a physical hand-drawn exemplar being created and captured simultaneously by another video camera. We believe interactive scenarios such as this will empower the creative potential of artists and inspire them with new creative ideas.

ACKNOWLEDGMENTS

We thank the reviewers for their insightful comments and suggestions. We are also grateful to Aneta Texler for her help on manuscript perparation as well as Zuzana Studená, Pavla & Jolana Sýkorová, and Emma Mitéran for participating in the recordings of live sessions. This research was supported by Snap Inc., the Research Center for Informatics, grant No. CZ.02.1.01/0.0/0.0/16_019/0000765, and by the Grant Agency of the Czech Technical University in Prague, grant No. SGS19/179/OHK3/3T/13 (Research of Modern Computer Graphics Methods).

REFERENCES

Pierre Bénard, Forrester Cole, Michael Kass, Igor Mordatch, James Hegarty, Martin Sebastian Senn, Kurt Fleischer, Davide Pesare, and Katherine Breeden. 2013. Stylizing Animation By Example. *ACM Transactions on Graphics* 32, 4 (2013), 119.

Eric P. Bennett and Leonard McMillan. 2005. Video Enhancement Using Per-Pixel Virtual Exposures. *ACM Transactions on Graphics* 24, 3 (2005), 845–852.

Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A. Efros. 2019. Everybody Dance Now. In *Proceedings of IEEE International Conference on Computer Vision*. 5933–5942.

Dongdong Chen, Jing Liao, Lu Yuan, Nenghai Yu, and Gang Hua. 2017. Coherent Online Video Style Transfer. In *Proceedings of IEEE International Conference on Computer Vision*. 1114–1123.

Zhuoyuan Chen, Hailin Jin, Zhe Lin, Scott Cohen, and Ying Wu. 2013. Large Displacement Optical Flow from Nearest Neighbor Fields. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 2443–2450.

Jakub Fišer, Ondřej Jamriška, Michal Lukáč, Eli Shechtman, Paul Asente, Jingwan Lu, and Daniel Sýkora. 2016. StyLit: Illumination-Guided Example-Based Stylization of 3D Renderings. *ACM Transactions on Graphics* 35, 4 (2016), 92.

Jakub Fišer, Ondřej Jamriška, David Simons, Eli Shechtman, Jingwan Lu, Paul Asente, Michal Lukáč, and Daniel Sýkora. 2017. Example-Based Synthesis of Stylized Facial Animations. *ACM Transactions on Graphics* 36, 4 (2017), 155.

Jakub Fišer, Michal Lukáč, Ondřej Jamriška, Martin Čadík, Yotam Gingold, Paul Asente, and Daniel Sýkora. 2014. Color Me Noisy: Example-based Rendering of Hand-colored Animations with Temporal Noise Control. *Computer Graphics Forum* 33, 4 (2014), 1–10.

Oriel Frigo, Neus Sabater, Julie Delon, and Pierre Hellier. 2019. Video Style Transfer by Consistent Adaptive Patch Sampling. *The Visual Computer* 35, 3 (2019), 429–443.

David Futschik, Menglei Chai, Chen Cao, Chongyang Ma, Aleksei Stoliar, Sergey Korolev, Sergey Tulyakov, Michal Kučera, and Daniel Sýkora. 2019. Real-Time Patch-Based Stylization of Portraits Using Generative Adversarial Network. In *Proceedings of the ACM/EG Expressive Symposium*. 33–42.

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. 2016. Image Style Transfer Using Convolutional Neural Networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 2414–2423.

Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. 2001. Image Analogies. In *SIGGRAPH Conference Proceedings*. 327–340.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 5967–5976.

Ondřej Jamriška, Šárka Sochorová, Ondřej Texler, Michal Lukáč, Jakub Fišer, Jingwan Lu, Eli Shechtman, and Daniel Sýkora. 2019. Stylizing Video by Example. *ACM Transactions on Graphics* 38, 4 (2019), 107.

Dmytro Kotovenko, Artsiom Sanakoyeu, Sabine Lang, and Björn Ommer. 2019. Content and Style Disentanglement for Artistic Style Transfer. In *Proceedings of IEEE International Conference on Computer Vision*. 4421–4430.

Wei-Sheng Lai, Jia-Bin Huang, Oliver Wang, Eli Shechtman, Ersin Yumer, and Ming-Hsuan Yang. 2018. Learning Blind Video Temporal Consistency. In *Proceedings of European Conference on Computer Vision*. 179–195.

Chuan Li and Michael Wand. 2016. Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks. In *Proceedings of European Conference on Computer Vision*. 702–716.

Xueting Li, Sifei Liu, Shalini De Mello, Xiaolong Wang, Jan Kautz, and Ming-Hsuan Yang. 2019. Joint-task Self-supervised Learning for Temporal Correspondence. In *Advances in Neural Information Processing Systems*. 317–327.

Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. 2019. Few-Shot Unsupervised Image-to-Image Translation. In *Proceedings of IEEE International Conference on Computer Vision*. 10551–10560.

Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. 2018. An intriguing failing of convolutional neural networks and the CoordConv solution. In *Advances in Neural Information Processing Systems*. 9628–9639.

Ravi Teja Mullapudi, Steven Chen, Keyi Zhang, Deva Ramanan, and Kayvon Fatahalian. 2019. Online Model Distillation for Efficient Video Inference. In *Proceedings of IEEE International Conference on Computer Vision*. 3572–3581.

Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. 2018. Artistic Style Transfer for Videos and Spherical Images. *International Journal of Computer Vision* 126, 11 (2018), 1199–1219.

Artsiom Sanakoyeu, Dmytro Kotovenko, Sabine Lang, and Björn Ommer. 2018. A Style-Aware Content Loss for Real-Time HD Style Transfer. In *Proceedings of European Conference on Computer Vision*. 715–731.

Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. 2019. SinGAN: Learning a Generative Model From a Single Natural Image. In *Proceedings of IEEE International Conference on Computer Vision*. 4570–4580.

Eli Shechtman, Alex Rav-Acha, Michal Irani, and Steven M. Seitz. 2010. Regenerative Morphing. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 615–622.

Assaf Shocher, Shai Bagon, Phillip Isola, and Michal Irani. 2019. InGAN: Capturing and Remapping the "DNA" of a Natural Image. In *Proceedings of IEEE International Conference on Computer Vision*. 4492–4501.

Assaf Shocher, Nadav Cohen, and Michal Irani. 2018. "Zero-Shot" Super-Resolution Using Deep Internal Learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 3118–3126.

Daniel Sýkora, John Dingliana, and Steven Collins. 2009. As-rigid-as-possible Image Registration for Hand-drawn Cartoon Animations. In *Proceedings of International Symposium on Non-Photorealistic Animation and Rendering*. 25–33.

Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky. 2016. Texture Networks: Feed-forward Synthesis of Textures and Stylized Images. In *Proceedings of International Conference on International Conference on Machine Learning*. 1349–1357.

Miao Wang, Guo-Ye Yang, Ruilong Li, Runze Liang, Song-Hai Zhang, Peter M. Hall, and Shi-Min Hu. 2019c. Example-Guided Style-Consistent Image Synthesis From Semantic Labeling. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 1495–1504.

Ting-Chun Wang, Ming-Yu Liu, Andrew Tao, Guilin Liu, Jan Kautz, and Bryan Catanzaro. 2019b. Few-shot Video-to-Video Synthesis. In *Advances in Neural Information Processing Systems*. 5014–5025.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018a. Video-to-Video Synthesis. In *Advances in Neural Information Processing Systems*. 1144–1156.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018b. High-Resolution Image Synthesis and Semantic Manipulation With Conditional GANs. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 8798–8807.

Xiaolong Wang, Allan Jabri, and Alexei A. Efros. 2019a. Learning Correspondence From the Cycle-Consistency of Time. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2561–2571.

Yang Zhou, Zhen Zhu, Xiang Bai, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. 2018. Non-stationary texture synthesis by adversarial expansion. *ACM Transactions on Graphics* 37, 4 (2018), 49.