Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Graphics and Interaction

Example-based Style Transfer of Hand-drawn Content

Ing. Ondřej Texler

A dissertation submitted to
the Faculty of Electrical Engineering, Czech Technical University in Prague,
in partial fulfilment of the requirements for the degree of
Doctor of Philosophy.

Ph.D. programme: Electrical Engineering and Information Technology
Branch of study: Information Science and Computer Engineering

Supervisor: prof. Ing. Daniel Sýkora, PhD

October 2020

Example-based Style Transfer of Hand-drawn Content

Ing. Ondřej Texler
texleond@fel.cvut.cz
Department of Computer Graphics and Interaction
Faculty of Electrical Engineering
Czech Technical University in Prague
Karlovo nam. 13, 121 35 Prague 2, CZ

Abstract

The topic of transferring an artistic style from a given example to another image has been very popular among the researchers over the last two decades. Although certain progress has been made, and computer-generated stylized images are becoming almost indistinguishable from real artworks, there are still fundamental challenges that need to be solved. In this dissertation, we focus on the task of Example-based Style Transfer and related methods, we discuss its applications and their state-of-the-art algorithmic solutions, and we follow by our own contribution into this field. In particular, our research contributes to the following areas: (1) example-based stylization methods allowing for a fast non-photorealistic rendering focusing on efficiency and high-quality results; (2) transferring an artistic style from an image to a video sequence where a semantically meaningful transfer is expected; (3) combination of patch-based and neural methods to allow for fully automatic style transfer while maintaining high-quality of the results. (4) research of learning based methods that allow high-quality semantically meaningful stylization of videos at interactive rates. Finally, we acquaint the reader with our vision of future work.

This thesis is presented as a collection of five research papers that were published in renowned impacted journals.

Keywords

Computer Graphics, Style Transfer, Example-based, Hand-drawn, Texture Synthesis, Stylization, Neural Networks, Non-photorealistic Rendering, Animation, Digital Art

Acknowledgements

First of all I would like to express my deepest gratitude to my advisor Daniel Sýkora for his kind guidance and persistent support throughout the doctoral studies, for his research ideas, and for both work-related and personal advice. I am grateful to my internship mentors, Michal Lukáč, Jakub Fišer, and Menglei Chai for taking care of me abroad and for being my role models. It goes without saying that I owe my thanks to all of my co-authors and collaborators for trusting in me and helping me with the research. I would like to give special thanks to the artists and others who helped to prepare visual material for our research, namely to, Pavla and Štěpánka Sýkorová, Zuzana Studená, and Jakub Javora. At last but not least, I would like to thank my wife Aneta and my parents for their faith in me and endless support.

The research presented in this thesis was conducted in collaboration with, and supported by, Adobe Research, Snap Inc., and Chaos Czech, and has further been supported by the Technology Agency of the Czech Republic under research program TE01020415 (V3C – Visual Computing Competence Center), by the Grant Agency of the Czech Technical University in Prague, grant No. SGS19/179/OHK3/3T/13 (Research of Modern Computer Graphics Methods), and by the Research Center for Informatics, grant No. CZ.02.1.01/0.0/0.0/16_019/0000765, by the Fulbright Commission in the Czech Republic, and by the Czech Science Foundation under project GA19-07626S.

Přenos ručně kresleného stylu na základě předlohy

Ing. Ondřej Texler
texleond@fel.cvut.cz
Katedra počítačové grafiky a interakce
Fakulta elektrotechnická
České vysoké učení technické v Praze
Karlovo náměstí 13, 121 35 Praha 2

Abstrakt

Po více než dvacet let se výzkumníci zabývají problematikou přenosu výtvarného stylu z dané předlohy na jiný obrázek. Ačkoli v této oblasti došlo k výraznému pokroku a obrazy generované počítačem vypadjí téměř jako reálná díla, stále jsou zde fundamentální překážky, které je třeba vyřešit. Tato disertační práce se zabývá přenosem výtvarného stylu na základě předlohy a dalšími metodami, které s tímto tématem úzce souvisí. Práce pojednává o aplikovatelnosti, state-of-the-art algoritmických řešeních a zahrnuje naši vlastní kontribuci do problematiky přenosu výtvarného stylu.

Náš výzkum přispívá zejména do následujících oblastí: (1) metody přenosu výtvarného stylu na základě předlohy umožňující nefotorealistické vykreslování s důrazem na výpočetní efektivnost a výsledky ve vysoké kvalitě. (2) přenos výtvarného stylu z obrázku na videosekvenci, kde je očekáváno zachování sémantiky. (3) kombinace patch-based a neurálních metod umožňujících přenos výtvarného stylu ve vysoké kvalitě a bez nutnosti interakce uživatele. (4) výzkum metod založených na hlubokých neuronových sítí s důrazem na efektivní trénink a generování výsledků v reálném čase a to při zachování vysoké kvality. Nakonec čtenáře seznámíme s naší vizí pro budoucí výzkum.

Tato práce je prezentována jako soubor pěti výzkumných článků, které byly publikovány v renomovaných impaktovaných časopisech.

Klíčová slova

Počítačová grafika, přenos výtvarného stylu, na základě předlohy, ručně kreslený, syntéza textury, stylizace, neuronové sítě, nefotorealistické vykreslování, animace, digitální umění

Contents

Chapter 1

Introduction

1.1 Motivation

Over the last three decades, creating of digital imagery has undergone tremendous development. Computational power as well as amount of available digital data increased exponentially; moreover, development of new algorithms and success of neural networks pushed the creating of digital art to the level unseen before. All these new digital painting tools and algorithms that become available allowed artists to be more efficient and productive. However, this rapid development brought new requirements on fidelity and quality of the digital imagery. Nowadays, FullHD or even 4K resolution images are considered standard, and producing and editing of such large images is more difficult for artists, and demands for new algorithms. Thanks to all these advances, the digital art has become very popular, and many artists and studios are trying to produce new artworks that were not possible before or were exceeding the available budget, e.g., fully hand-drawn ninety minutes movie or fully stylized computer game. This increased demand brings even more requirements and motivates the further research. The algorithms are expected to be faster, to work on arbitrary resolution, and to work as autonomously and intelligently as possible. Motivated by aforementioned facts, this thesis focus on important and thriving part in creating of digital art, non-photorealistic rendering and style transfer.

In the field of computer graphics, non-photorealistic rendering is a well established area of research. Opposed to photorealistic rendering where the goal is to create realistically looking images respecting propagation of light in the scene, the goal of non-photorealistic rendering is to create stylized imagery, e.g., hand-drawn paintings, cartoons, or sketches. In our work, we focus mainly on a special subtask of non-photorealistic rendering—style transfer; the task where an artistic style is transferred from an exemplar image to another image or video.

In this chapter, we first briefly introduce a style transfer, its early as well as more advanced applications, and we mention algorithms and techniques to solve this task. Lastly, we mention the structure of this thesis.

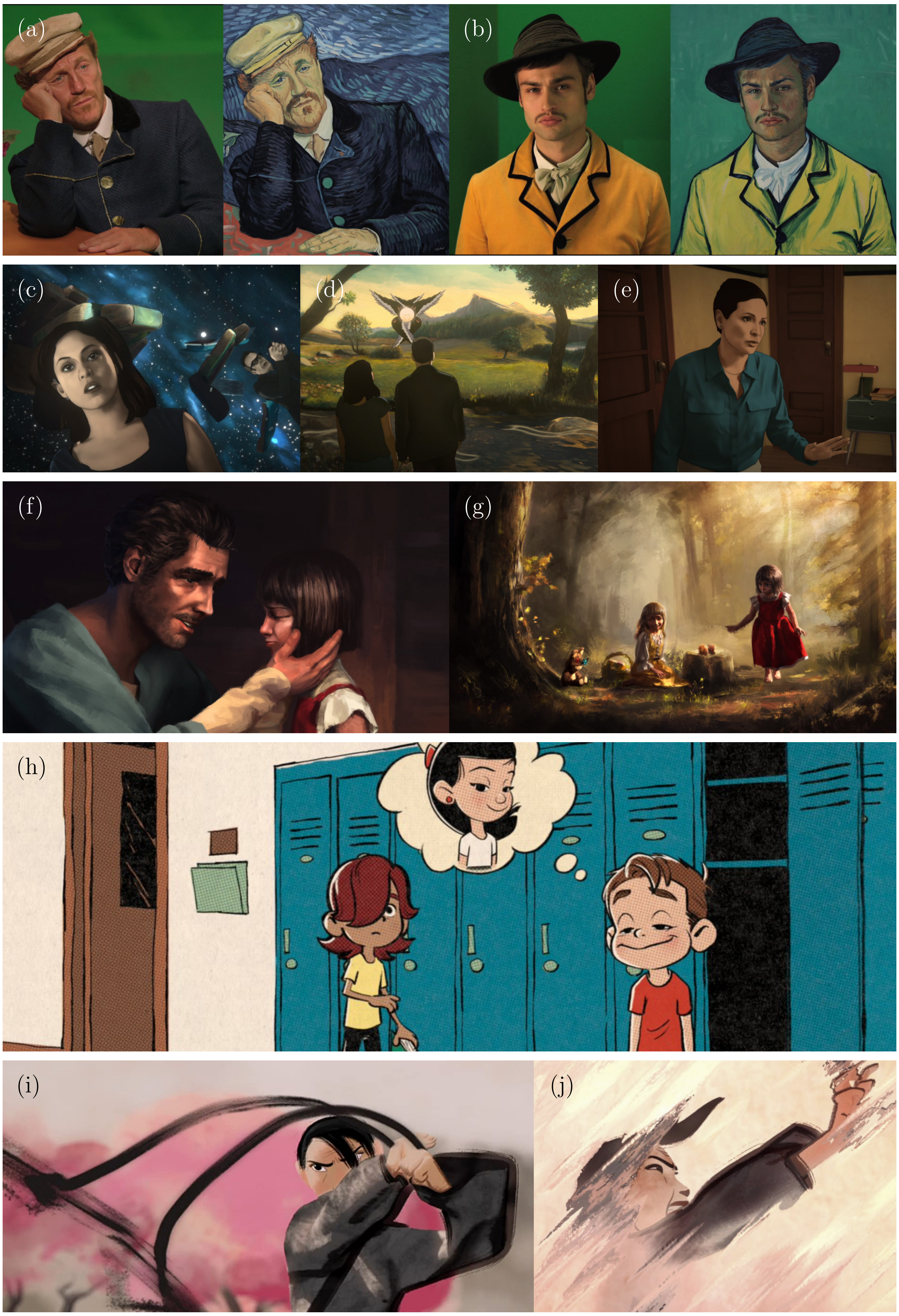


Figure 1.1: *Example of recently published fully stylized movies. Hand-painted movie Loving Vincent (a-b), where original live captured footage is shown together with its painted variant. Animated movie Undone (c-e) by Amazon, painterly short film Annie (f-g) by Riot Games, and short animated movies Just a Thought (h) & Jing Hua (i-j) by Disney. (All images come from respective movie trailers or behind-the-scenes documentaries.)*

1.2 Brief Introduction into the Style Transfer

Broadly speaking, the goal of style transfer is to create images resembling real artworks; usually, transfer an artistic style from a painting to an image depicting a different content. Style transfer methods can be used in various areas, for example in movie production (e.g., The Little Mermaid–1989 or the original Beauty and the Beast–1991 by Disney) where creating of traditional hand-drawn animations is usually extremely labor intensive as every frame has to be painted or modified individually by an artist. The technique commonly used in animation is called rotoscoping; a live-action scene is shoot, and an artist then paints over every frame which creates the final hand-drawn animation. Since this is truly tedious, work there are not many full-length movies created using this technique. Although recently, this technique was used to create two long movies, *Loving Vincent*[1], where every frame was painted by hand and it took a large team six years to finish the movie; and *Undone*[2] which took an enormous human effort as well. And having a style transfer technique that does not require effortful user interaction and is capable of producing high visual quality results could make the process of creating these movies tremendously easier. In the Fig. 1.1, see the images from the mentioned movies together with recently presented painterly short film *Annie*[3] created by Riot Games, and stylized short movies *Just a Thought*[4] & *Jing Hua*[5] by Disney.

Over the last two decades, development of new algorithmic solutions, new hardware capabilities, and advances made in neural networks allowed the field of style transfer to progress; from early concepts [Her+01], over tools appreciated by artistic community [GEB16][6] or [JAFF16][7], to applications in AR [Mag+15]. The mentioned progress is briefly described in the following paragraphs and then more thoroughly in the related work chapter.

Predecessors of style transfer were based on domain-specific algorithmic solutions. They used a physical simulation to reproduce the behavior of a given artistic medium, e.g., simulation of watercolor and the optical effect of the superimposed glazes [Cur+97]. Procedural solution to hallucinate a particular artistic style [Bou+06; Bou+07]. Or they composed the desired content from a set of predefined elements, e.g., brush-strokes [Her98]; more advanced approach in the similar spirit RealBrush [Lu+13], was later developed; in this case example strokes are given and the algorithm uses them to prepare the final result.

The important milestone opening an era of example-based style transfer has been reached by introducing *Image Analogies* [Her+01]; an approach where no domain is assumed and the only input to the method is an example of stylization – original image and its stylized version. The framework is than able to stylize other images in the same way as the exemplar stylization was performed.

A novel approach to the style transfer was proposed by [GEB16]; they used convolutional neural networks to perform a style transfer. The result images are created by

[1]Loving Vincent: http://lovingvincent.com (Accessed October 11, 2020)
[2]Undone: https://www.imdb.com/title/tt8101850/ (Accessed October 11, 2020)
[3]Annie: https://www.youtube.com/watch?v=aUTU-GnxVuM (Accessed October 11, 2020)
[4]Just a Thought: https://www.imdb.com/title/tt10229080/ (Accessed October 11, 2020)
[5]Jing Hua: https://www.imdb.com/title/tt10229066/ (Accessed October 11, 2020)
[6]DeepArt: https://deepart.io (Accessed October 11, 2020)
[7]Prisma: https://prisma-ai.com (Accessed October 11, 2020)

finding an image that simultaneously matches the content representation of the photograph and the style representation of the artwork in the domain of VGG [SZ14]. Due to its optimization nature, their approach is slow and hardware demanding; and due to the fact the style image is in the process represented by set of statistics, the results are usually blurry lacking fine artistic details). However, they opened a completely new path in the field of style transfer and inspired many other researchers.

The natural extension of transferring style from one image to another image is to transfer artistic style to entire video sequences. The problem there is time coherency. When style transfer is performed using previous methods individually frame-by-frame, the resulting video will most likely heavily flicker. This issue has been addressed by many researchers, for example [Che+17].

1.3 Structure of the Thesis

In this chapter we motivated our research, and briefly presented the task of style transfer. In Chapter 2 we discuss related work, and in Chapter 3 we briefly overview contributions of this dissertation. Following, in Chapters 4, 5, 6, 7, and 8 we describe our individual contributions in more depth. Each mentioned chapter corresponds to a publication that was published in a journal with impact factor. In Chapter 9 we summarize contribution of our work, briefly mention concurrent work, and conclude the thesis with our vision for future work.

Appendix A lists author's publications together with a list of their citations to date in other publications. Appendix B specifies author's contribution to the individual papers presented in this thesis.

Finally, Appendix C contains supplementary material for our work [Jam+19], Appendix D contains supplementary material for [Tex+20b], and Appendix E contains additional material for StyleBlit [Hau+20].

Chapter 2

Related Work

In this chapter we describe different research paths and recent work in the field of style transfer. We begin with early methods that perform image stylization by making a composition of predefined elements (typically brush strokes). We continue with more general example-based methods, where an entire painting is given as an example. Finally, we introduce recent methods based on neural networks and their combinations with algorithmic approaches. We also briefly discuss state-of-the-art in video stylization that consider temporal coherence.

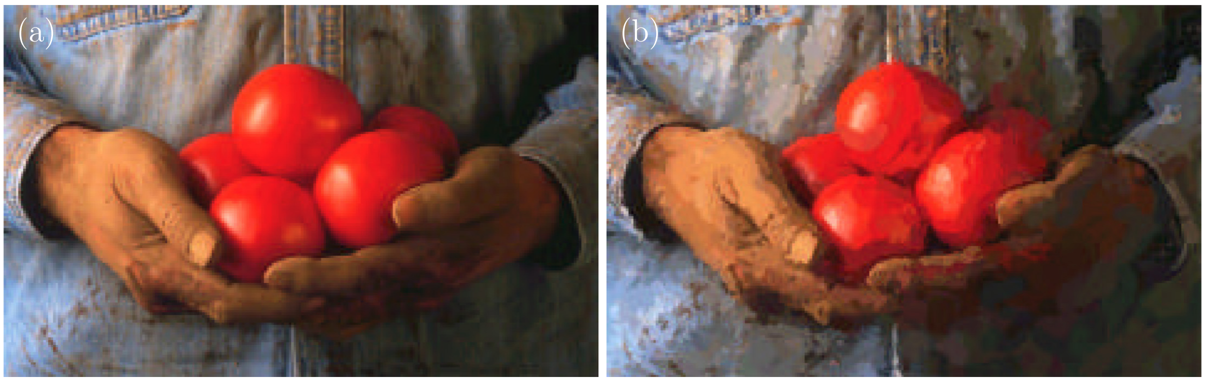


Figure 2.1: *Early method to the painterly rendering [Her98]. A source image (a) depicts the content to be painted. Image (b) is the result obtained by applying small radius brushes over the source image (a). They apply brushes in multiple layers; large brushes are applied first, medium and small brushes follow.*

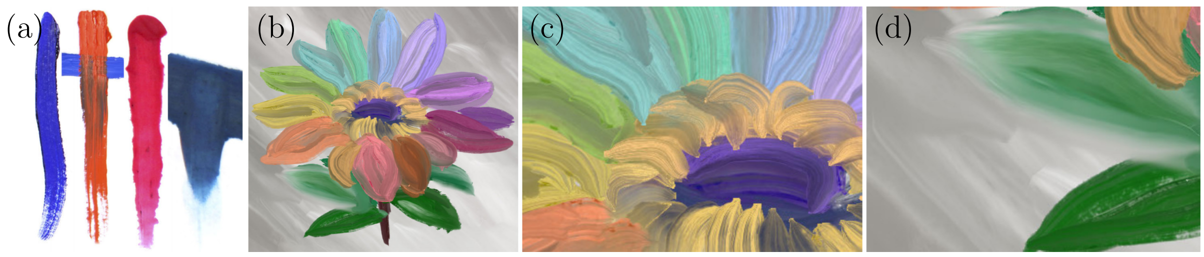


Figure 2.2: *RealBrush [Lu+13]. Brush stroke exemplars (a) are used to synthesize the painting (b). The foreground flower strokes, see close-up (c), come from oil exemplars(a–left), while the background strokes, see close-up (d), are composed using plasticine exemplars (a–right).*

Algorithmic and Stroke-based

One of the first methods that enabled generation of stylized imagery or videos were procedural techniques that rely on hand-crafted algorithmic solutions to reproduce a particular artistic style. Some of them, in order to mimic behavior of the given artistic medium, utilize physical simulation [Cur+97; Hae+07; LXJ12] or a procedural solution [Bou+06; Bou+07; Bén+10; Mon+18]. Other, so called stroke-based approaches use a library of predefined strokes that are rotated and translated according to some guidance information (e.g., segmentation or direction of image gradients). This approach has been applied both in 2D [Her98] as well as in 3D [Sch+11]; and in various artistic styles such as pen and ink illustration [Sal+97; Pra+01; Sna+06], hatching [Bre+07], or brush strokes [Lit97; HE04; Sch+11; ZZ11]. See one example in Fig. 2.1, where a photograph is composed from different brush strokes. The main drawback of this line of work is the fixed sets of strokes, textures, or patterns that are not allowing for a greater output variety. This limitation is partly overcome by introducing example brushes [Lu+13], see an overview in Fig. 2.2, and later in [Zhe+17].

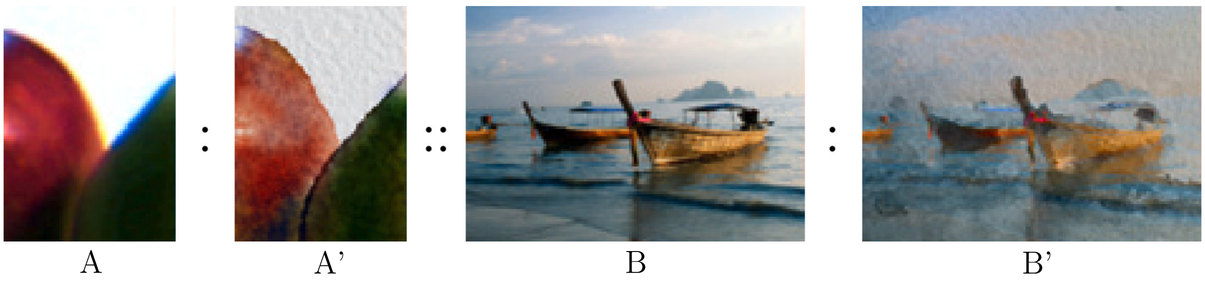


Figure 2.3: *Image Analogies [Her+01]. An unfiltered image A together with its filtered–stylized version A′ define the transformation. The goal is to perform the same transformation on an unfiltered image B to obtain its filtered version B′. Besides A, A′, and B, there is no other input to the framework.*

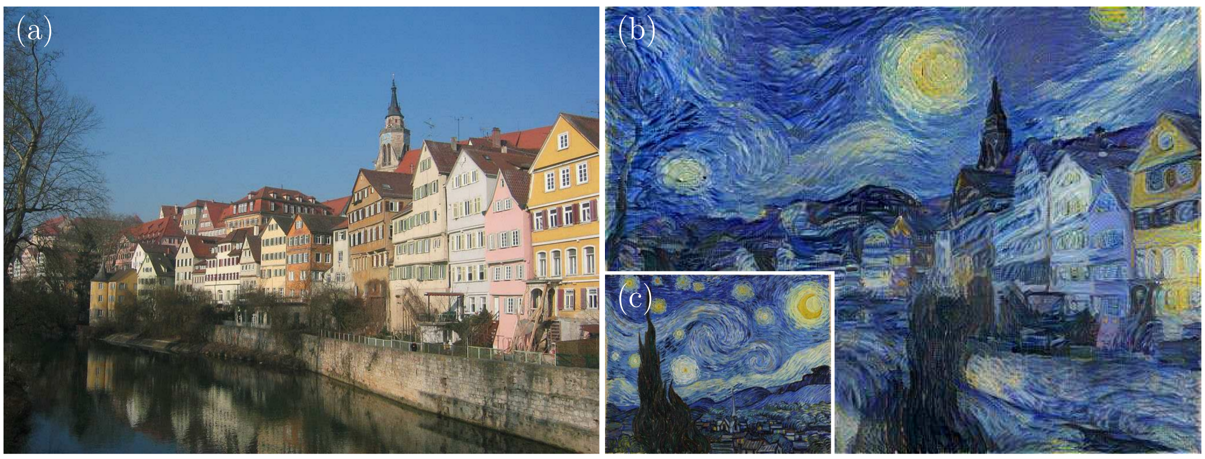


Figure 2.4: *An example of a novel neural network based approach to the style transfer [GEB16]. Result image (b) depicts the same content as source photograph (a) and bears artistic attributes of style exemplar (c).*

Example-based

To address limitations of approaches that are based on an algorithmic solution or on a set of predefined brushstrokes or textures, more general example-based approaches were developed; in these methods the style is given by an exemplar image.

(1) *Image Analogies* framework introduced by Hertzmann et al. [Her+01] allows arbitrary style transfer from a source exemplar to a target image using guided patch-based synthesis. See the method overview in Fig. 2.3. Image A is the original exemplar and image A' is its stylized counterpart, and these two images define the transformation. The task is then to apply this transformation to another image B in order to get image B' stylized in the same way the image A' was stylized. (2) *The Lit Sphere* approach by Sloan et al. [Slo+01] that uses texture mapping to generate stylized shading. (3) *Neural approaches* that use deep convolutional networks to perform style transfer, see example in Fig. 2.4. And lastly (4), we mention hybrid approaches where patch-based synthesis methods are combined with neural ones.

Patch-based Approaches

In Image Analogies [Her+01] guided patch-based synthesis was used to perform style transfer. The underlying patch-based machinery was further improved in other publications [WSI07; Kas+15; Fi16; BKR17]. Others adopted this framework to various stylization scenarios such as fluid animation [Jam+15], 3D renders [Fi16], facial animations [Fi17], or videos [Fi14; Dvo+18] where temporal coherence was formulated as an additional guidance channel for patch-based synthesis. Mentioned methods share one critical drawback, and it is preparation of custom-tailored guiding channels which are necessary to deliver compelling stylization quality. Another possible drawback is that applying patch-based synthesis at a higher resolution requires significantly more computational power thus makes mentioned methods hardly accessible in interactive scenarios. These two issues prevent artists from using style transfer efficiently in practice.

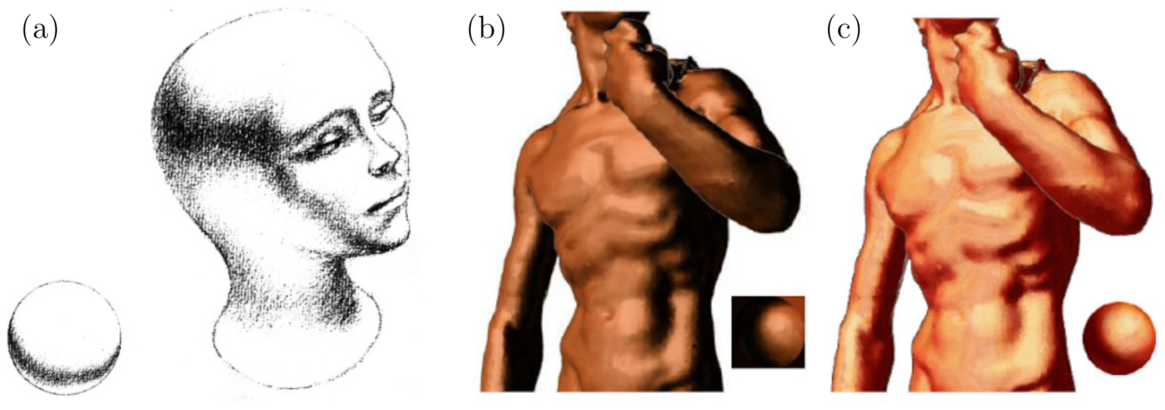


Figure 2.5: *An example of The Lit Sphere [Slo+01]. The goal here is to transfer a shading study from a simple sphere to a complex 3D model. (a) shading is transferred to a head model. (b-c) a sphere serves as a shading reference to render the body model.*

MatCap

Sloan et al. [Slo+01] introduced technique called *The Lit Sphere*, see the example in Fig. 2.5 – today, this concept is in 3D artist community known as MatCap (which stands for material capture). *The Lit Sphere* was later extended by others [BTM06; TAY13]. The main idea here is to uses a one-to-one correspondence between normal values to transfer style from a hand-drawn exemplar of a simple object (a sphere) to a more complex 3D model. This can be also solved using environment mapping [BN76]. Although the mentioned approaches can perform style transfer in real-time, they work well only when the style exemplar does not contain distinct high-frequency details – these details would be affected by geometric distortions caused by texture-mapping algorithm and thus compromise fidelity of results. MatCap is used in most of the 3D artistic software nowadays, and the mentioned limitation significantly reduces the number of usable materials, thus leaves artists with no other option than using simple, low-frequency, textures.

Neural-based

Recently, Gatys et al. [GEB16] pioneered the use of neural networks to solve the task of style transfer. They employ pre-trained convolutional neural network VGG [SZ14], which is trained for image classification, to match the statistics of both the style and content images to create result having desired content as well as stylized look; this approach bears resemblance to parametric texture synthesis [PS00]. The same authors further extended their idea to allow control over spatial location, color information, and scale of features [Gat+17]; and others [SID17] extended this technique as well. Major drawback common to mentioned neural techniques is that they are usually unable to preserve high-frequency details of the style exemplar; it is caused by their statistics-based nature. Moreover, since they use back-propagation mechanism to refine the final result in multiple iterations, they are computationally demanding, and do not allow interactive response even if run on GPU. To address the performance issue pre-trained feed-forward networks [JAFF16; Uly+16a; DSK16; Che+17] were utilized to speed up the stylization process significantly.

A modern take to generate images or videos is to employ generative adversarial networks [Goo+14]. By utilizing this concept, many so-called image-to-image [Iso+17; Zhu+17a; Zhu+17b; Tul+18; Wan+18b] and video-to-video appearance translation networks [Wan+18a; Cha+19] were developed, and can be applied to the style transfer task. However, crucial drawback here is that a huge training dataset is required, and every new style requires additional, usually costly, training. To overcome necessity for additional training, encoder–decoder scheme was proposed [Li+17; HB17; Lu+17]. Encoder, usually convolutionals layers of VGG [SZ14], is used to represent both the style and content image as a set of feature channels. These features are then combined and pre-trained decoder turns mentioned features back into the image. However, the problem of lacking high-frequency details remains in this case as well. Recently, a complex encoder–decoder system that can reproduce the details well was proposed by Kotovenko et al. [Kot+19b; Kot+19a]. However, this approach is still not able to deliver semantically meaningful results.

To bypass the need for large dataset of a specific style, few-shot learning methods [Wan+19a; Wan+19b] that can perform appearance translation were recently presented. However, these methods has to be pre-trained on a large dataset of domain-specific training data (e.g., human bodies in motion or facial videos). Thus, these techniques cannot be used if the target domain is not known beforehand.

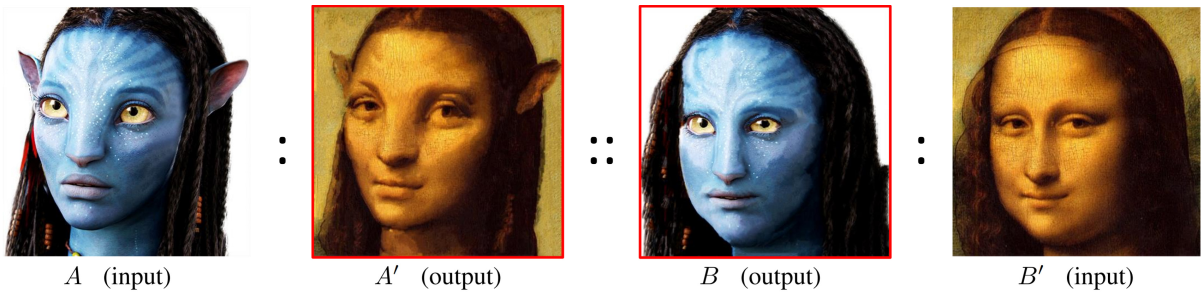


Figure 2.6: *Hybrid approach Deep Image Analogy [Lia+17]. Here, guidance channels that are used in the original Image Analogies [Her+01] are replaced by responses from the VGG [SZ14] image recognition network. This leads to convincing results (images A' and B), however, the input images (A and B') has to depict the same content, e.g., face.*

Hybrid: Patch-based Combined with Neural-based

Recent research showed, that promising direction is to combine neural networks with patch-based synthesis methods. First, Li et al. [LW16a] searches local neural patches from style image concerning the structure of a content image, and is able to achieve better reproduction of local textures. Next, neural version of Image Analogies [Her+01] called Deep Image Analogy [Lia+17] is introduced, see Fig. 2.6. Recently, reshuffle in the spirit of [Kas+15] is proposed to overcome a problem of extensive use of particular features [Gu+18]. Lastly, Futschik et al. [Fut+19] proposes patch-based method [Fi17] to generate training dataset of stylized portraits and then train adversarial neural network to perform style transfer.

Temporal Consistency

Temporal consistency, i.e., flickering between individual video frames, is an important aspect that needs to be taken into account when performing style transfer to animation or video. When every frame is stylized individually, once played as an animation, there is likely to be an intense temporal noise, see Fig. 2.7. In some cases, the flickering is desired effect [Fi14] as it is natural for traditional hand-colored animations. However, in certain scenarios it is compulsory to control or entirely suppress any flickering. One way to enforce the temporal consistency is to consider previous frame when the new frame is stylized; this was done in the patch-based [Bén+13; Fi17; Dvo+18; Jam+19; Fri+19] as well as in neural-based domain [Che+17; Gup+17; San+18; RDB18]. Another way is to use blind temporal coherency approach [Lai+18] in the post-processing step; it takes per-frame stylized video as input and outputs a temporally consistent video.

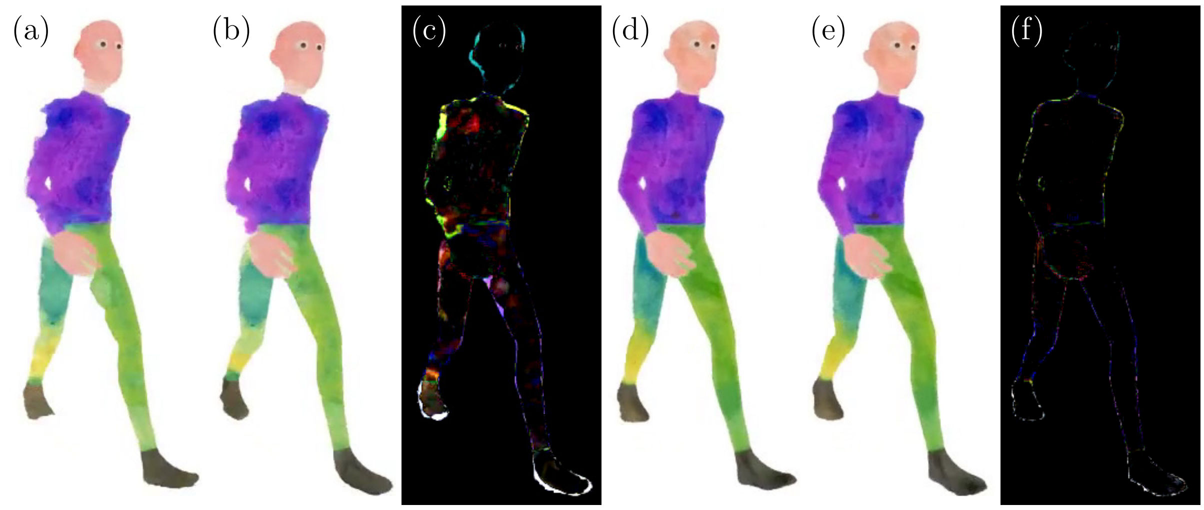


Figure 2.7: *Overview of temporal consistency. Images (a-b) are two consecutive frames that were stylized independently, (c) is their pixelwise difference considering motion compensation, black areas depicts zero difference. Images (d-e) are two consecutive frames stylized with temporal coherency, (f) is their pixelwise difference considering motion compensation. Although both sequences (a-b) and (d-e) look identical, their respective differences are different. (c) contains large non-zero areas inside the body, that means that if you play the sequence as an animation, significant flickering will occur at these locations. The image (f) is mostly black (i.e., zero difference), the animation will not flicker.*

Summary

Publications mentioned in this chapter serve as an overview of research in the field of style transfer and related disciplines; it clearly shows the direction in which the related research is going, and it gives us a motivation and hints on which topics to focus. Next in this thesis, we present five publications that are closely related or extend the aforementioned approaches.

Chapter 3

Our Contribution

The main goal of our research is to develop new algorithms and methods that help artists as well as casual users to create painted content, experiment more efficiently, and save time by automating repetitive work; and enable visually rich hand-drawn experience in applications where it was not previously possible. We want to explore and develop new optimization methods that can be used to solve highly intuitive tasks, e.g., paint an image indistinguishable from the one painted by a human. We focus mainly on example-based style transfer, the task where an example of the artistic style is given in the form of a painting. The goal is then to create images with different content while maintaining the artistic look of the given example. Although it is highly subjective to tell which attributes of artwork are part of the *style* and which are part of the *content*; as a style, we usually consider properties of brush strokes (e.g., rotation or size), used artistic medium (e.g., watercolor, acrylic paint, pastels, etc.), and attributes of canvas structure. In the following paragraphs we briefly overview contributions of this dissertation. More in-depth discussion can be found in the main part of this thesis.

3.1 Neurally-Guided Patch-Based Synthesis

An ultimate goal of style transfer research is to solve a problem of general style transfer—a case where no domain is assumed, i.e., both a style exemplar and a target image are 2D images depicting arbitrary content. See Fig. 3.1, where a style from the painting of a landscape is transferred to the photograph of a horse. In this general case, it is usually hard to establish meaningful correspondences between the style exemplar and target content; thus it is hard to guide style transfer method to produce results inline with the artist's intention.

In Chapter 4 we introduce our solution for this problem. We present a new approach to example-based style transfer combining neural methods with patch-based synthesis to achieve compelling stylization quality even for high-resolution imagery. We take advantage of neural techniques to provide adequate stylization at the global level and use their output as a prior for subsequent patch-based synthesis at the detail level. Thanks to this combination, our method keeps the high frequencies of the original artistic media better, thereby dramatically increases the fidelity of the resulting stylized imagery. We show how to stylize extremely large images (e.g., 340 Mpix) without the need to run the synthesis at the pixel level, yet retaining the original high-frequency details. We demon-

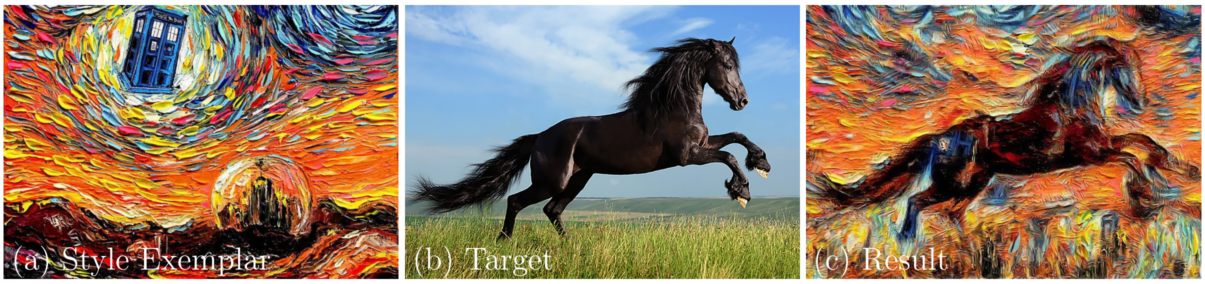


Figure 3.1: *Our approach [Tex+20a] used to solve the task of generic example-based style transfer. The goal here is to transfer artistic style from the given style exemplar (a) to the given target (b) while preserving the appearance of (a) and the content of (b); result of our method is (c). Painting (a) courtesy of ©Aja Kusick.*

strate the power and generality of this approach on a novel stylization algorithm that delivers comparable visual quality to state-of-art neural style transfer while completely eschewing any purpose-trained stylization blocks and only using the response of a feature extractor as guidance for patch-based synthesis.

3.2 Stylizing Video by Example

Another task of example-based style transfer we focus on in our research is a stylization of videos. To precisely follow an artist's intention, the task is solved within the domain of a single video sequence where meaningful correspondences can be established. See the overview of single-video sequence stylization in Fig. 3.2. Input is a video sequence and one keyframe – by artist pained frame from the original sequence. The task is then to propagate an artistic style from keyframes to the rest of the sequence while preserving visual quality and fine artistic details such as brush strokes or used artistic medium (e.g., watercolor, acrylic paint, etc.).

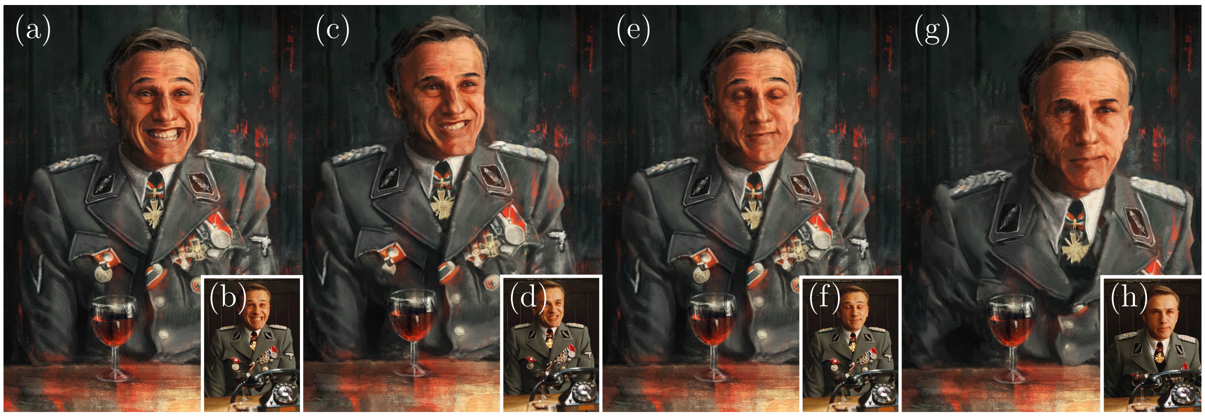


Figure 3.2: *Our method [Jam+19] used to stylize a video sequence. Assume (b, d, f, g) is the video sequence to be stylized. Artist takes one frame from this sequence (b) and provides its stylized version (a). The goal is then to propagate the artistic style from (a) to the rest of the sequence (c, e, g). Result of our method is (c, e, g). Painting (a) courtesy of ©Alice X. Zhang. The underlying footage is taken from the Inglourious Basterds movie.*

In Chapter 5, we introduce a new example-based approach to video stylization, with a focus on preserving the visual quality of the style, user controllability and applicability to arbitrary video. Our method gets as input one or more keyframes that the artist chooses to stylize with standard painting tools. It then automatically propagates the stylization to the rest of the sequence. To facilitate this while preserving visual quality, we developed a new type of guidance for state-of-art patch-based synthesis, that can be applied to any type of video content and does not require any additional information besides the video itself and a user-specified mask of the region to be stylized. We further show a temporal blending approach for interpolating style between keyframes that preserves texture coherence, contrast and high frequency details. We evaluate our method on various scenes from real production setting and provide a thorough comparison with prior art.

3.3 Stylization Using Few-Shot Patch-Based Training

Solutions based on deep neural networks bring significant advantage over the traditional optimization methods in many aspects. Thanks to heavy research and wide availability of highly optimized deep learning frameworks (e.g., PyTorch or TensorFlow), and optimized GPU architectures (e.g., NVIDIA's tensor core or Google's Tensor Processing Unit), the neural networks are practical and accessible to studios as well as individual artists. With this in mind, we extended the usecase presented in our previously discussed contribution [Jam+19], and developed a neural network based video stylizing tool that is able to perform the style transfer in real-time, and allows for an interactive scenario that was not possible before, see Fig. 3.3. Moreover, it overcomes other crucial drawbacks of concurrent neural based methods, e.g., training speed, required amount of training data, and it delivers results that are semantically meaningful.

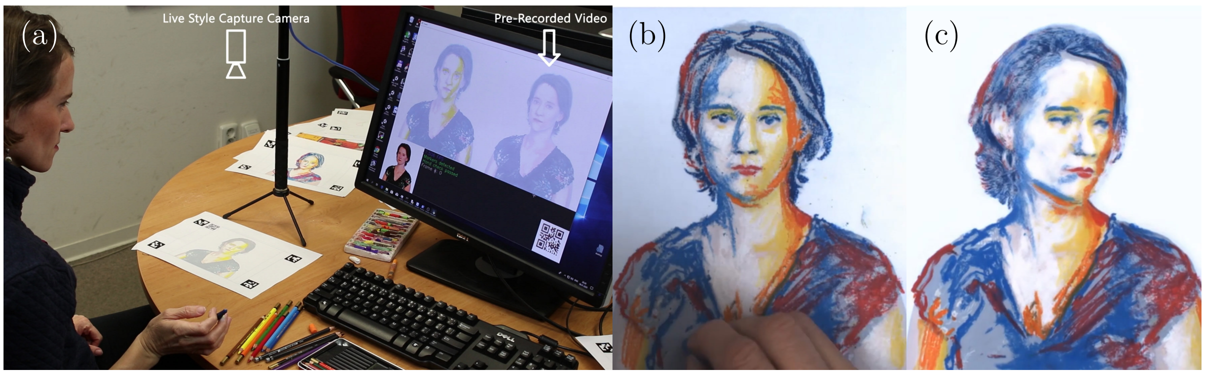


Figure 3.3: *Our method [Tex+20b] is used in the following scenario. The artist (a) paints over the image printed on a prepared stencil. While she is painting, the camera captures her work (b), the image-to-image translation network is learned to reproduce the artwork on the fly, and the video sequence (c) is stylized in real-time. Painting (b) courtesy of ©Zuzana Studená.*

In Chapter 6, 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, this 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 preserving temporal consistency without the need to stylize previous frames. 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.

3.4 StyleBlit: Stylization with Local Guidance

Our research also contributes to the field of non-photorealistic 3D rendering. The goal, in this case, is to artistically paint a 3D object to hallucinate a hand-drawn look; and the requirement is for it to run in real-time. Our setting is shown in Fig. 3.4; the task is to transfer a style from a simple 3D object to a more complex 3D object. Our method brings significant quality improvements over the concurrent techniques while maintaining high performance.

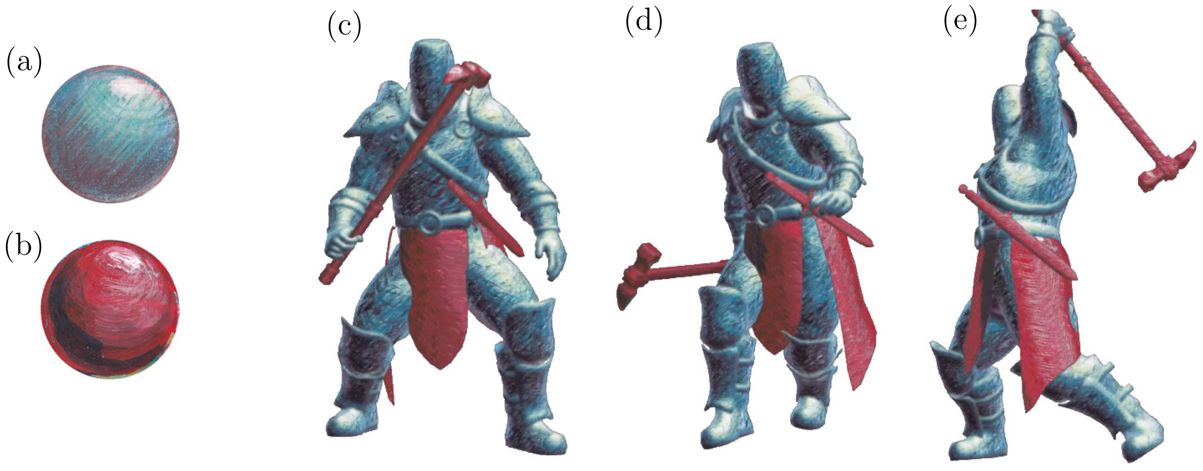


Figure 3.4: *Example of our method StyleBlit [Sýk+19]. The goal here is to transfer the style from a simple 3D objects–spheres to a more complex 3D object–knight, and the whole process is required to run in real-time in a shader. In this case, style from the sphere (a) is transferred to knight's armor and style from the sphere (b) is transferred to knight's accessories. Paintings (a, b) courtesy of ©Pavla Sýkorová.*

In Chapter 7, we present our contribution into this field, StyleBlit—an efficient example-based style transfer algorithm that can deliver high-quality stylized renderings in real-time on a single-core CPU. Our technique is especially suitable for style transfer applications that use local guidance - descriptive guiding channels containing large spatial variations. Local guidance encourages transfer of content from the source exemplar to the target image in a semantically meaningful way. Typical local guidance includes, e.g., normal values, texture coordinates or a displacement field. Contrary to previous style transfer techniques, our approach does not involve any computationally expensive optimization. We demonstrate that when local guidance is used, optimization-based techniques converge to solutions that can be well approximated by simple pixel-level operations. Inspired by this observation, we designed an algorithm that produces results visually similar to, if not better than, the state-of-the-art, and is several orders of magnitude faster. Our approach is suitable for scenarios with low computational budget such as games and mobile applications.

3.5 StyleProp: Stylization of 3D Models

Crucial requirement of real-time non-photorealistic rendering of 3D methods (e.g., our previous contribution StyleBlit [Sýk+19] is their speed. To satisfy this requirement, certain trade-offs in quality are usually made. StyleBlit is fast, but it requires local guidance and it is prone to certain artifacts (see the Limitation section in Chapter 7). To allow full stylization experience that can be delivered by computationally expensive optimization based methods, we propose a framework that seamlessly interpolates between the pre-computed stylized results, thus maintaining low requirement for computational budget, see Fig. 3.5.

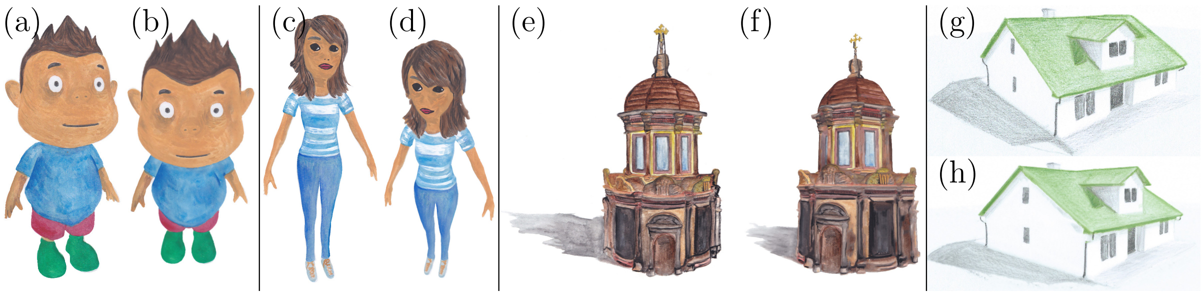


Figure 3.5: *Given a single hand-painted example (a, c, e, g), our method StyleProp [Hau+20] is able to faithfully stylize the same 3D model in different viewpoints (b, d, f, h). And thanks to the pre-compute step the stylizations are delivered in high-quality in real-time. Paintings (a, c, e) courtesy of ©Štěpánka Sýkorová, painting (g) courtesy of ©Barbora Kociánová.*

In Chapter 8, we present a novel approach to the real-time non-photorealistic rendering of 3D models in which a single hand-drawn exemplar specifies its appearance. We employ guided patch-based synthesis to achieve high visual quality as well as temporal coherence. However, unlike previous techniques that maintain consistency in one dimension (temporal domain), in our approach, multiple dimensions are taken into account to cover all degrees of freedom given by the available space of interactions (e.g., camera rotations). To enable interactive experience, we precalculate a sparse latent representation of the entire interaction space, which allows rendering of a stylized image in real-time, even on a mobile device. To the best of our knowledge, the proposed system is the first that enables interactive example-based stylization of 3D models with full temporal coherence in predefined interaction space.

Chapter 4

Neurally-Guided Patch-Based Synthesis

4.1 Introduction

In recent years, advances in neural style transfer and guided patch-based synthesis made the field of computer-assisted stylization very popular. Various publicly available software solutions (see, e.g., Prisma [JAFF16], DeepArt [GEB16], StyLit [Fi16], FaceStyle [Fi17]) successfully brought the style transfer concepts to consumers. These applications enjoy popularity among casual users due to their novelty factors. However, they are not addressing the needs of professional users who demand high-resolution, high-quality output accurately preserving the textural details of the original artistic exemplar.

Though guided patch-based synthesis approaches [Fi16; Fi17] can meticulously preserve fine-grained details, they require preparation of guidance channels. These guidance channels are important for establishing meaningful correspondences between the target image and the source style exemplar. Previous work designed guidance channels for specific use cases such as faces [Fi17], but designing meaningful guidance automatically in general case remains a difficult problem. On the other hand, neural-based style transfer [GEB16; Gu+18] does not require explicit guidance to produce good stylization effects at a global level. Nevertheless, due to its convolutional nature, it usually fails to preserve low-level details such as brush strokes or canvas structure that are important to retain the fidelity of the underlying artistic media.

Neural techniques are also limited to work at lower resolutions (typically below 1K), which does not suit the need for FullHD, 4K or higher resolution used in real production settings. A similar limitation also holds for guided patch-based synthesis where the processing time grows significantly with increasing output resolution. Neural style transfer algorithms also have the problem of exhausting GPU memories where going beyond 4K resolution becomes impossible under current hardware constraints.

In this paper, we propose a straightforward approach which overcomes the aforementioned limitations by combining neural style transfer, patch-based synthesis, and dense correspondence field upscale. We first apply neural style transfer to obtain semantically meaningful stylization at a global level without the need of user intervention, and then use patch-based synthesis to remove low-level artifacts and restore the color and fine details to retain the fidelity of the original style, see Fig. 4.2. To significantly reduce

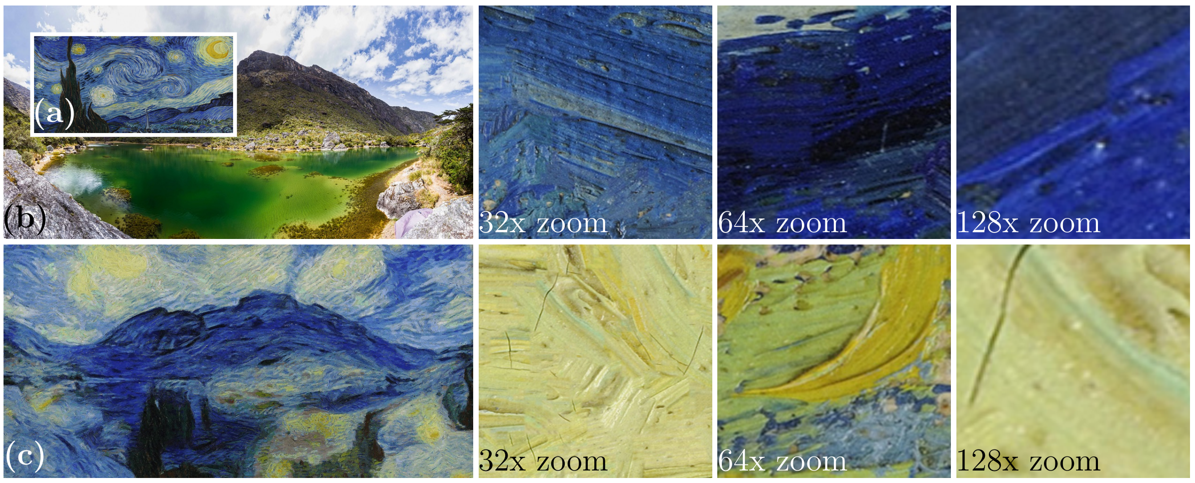


Figure 4.1: *An example of stylizing an extremely high-resolution image using our proposed method: (a) style exemplar of* 26400×13100 *px, (b) content image of the same resolution, (c) low resolution result of [GEB16] enhanced and enlarged by our method to the mentioned resolution. To the right, zoom-in patches of different parts of (c) up to zoom of 128× are shown; see all the individual brush strokes and its sharp boundaries. Also, notice how the structure of the original canvas and little cracks of the painting are preserved.*

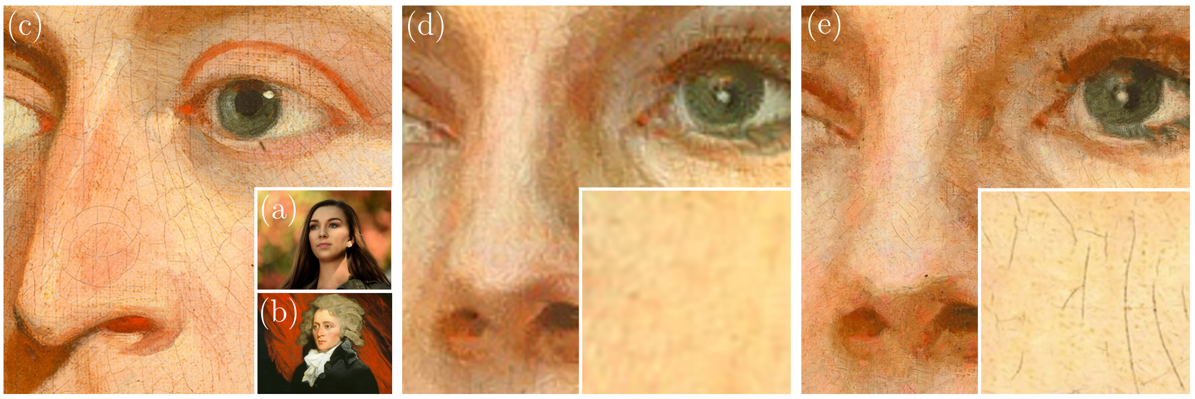


Figure 4.2: *An example of enhancing the result of neural-based approach using our method: (a) target photograph, (b) style exemplar of the same size, (c)* 6× *zoom-in to the style exemplar, (d) the output of neural-based method DeepArt [GEB16] is capable to perform convincing stylization; nevertheless, the image contains artifacts caused by the parametric nature of the used neural network. High-frequency details like the structure of strokes and canvas are largely lost, sacrificing the visual quality of the original artistic medium. In contrast, our method (e) brings significant quality improvement, it restores the individual brush strokes and boundaries between them faithfully, the result better reproduces the used artistic medium as well as canvas' structure. Note how the cracks of the original artwork are preserved; although zoom-in patches are shown, we encourage the reader to zoom-in even further.*

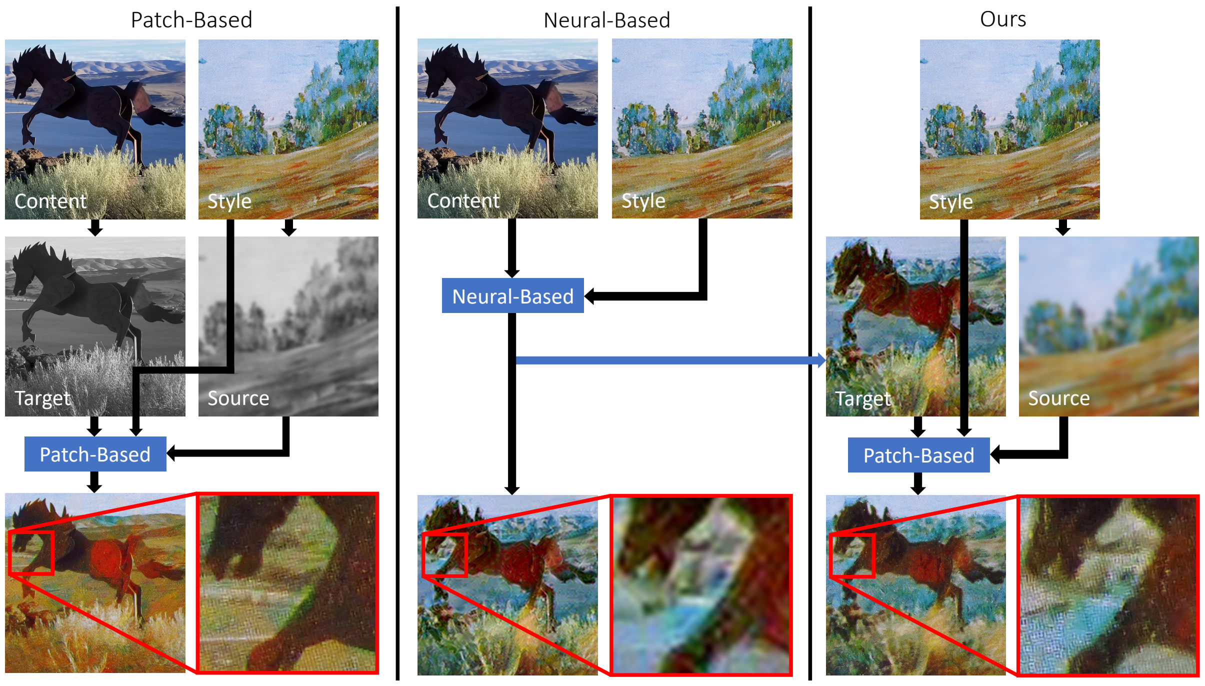


Figure 4.3: *Simplified scheme of a patch-based, neural-based, and our hybrid style transfer method: The left column shows a patch-based approach [Fi16] with guidance based on blurred grayscale images as proposed in the original Image Analogies method [Her+01]. The resulting image has high texture quality and preserves artistic attributes and canvas structure well; however, the result does not properly respect the content semantics, causing water to become brown. The middle column shows a neural-based approach [GEB16], no guidance channels are needed and global style properties and image semantic are preserved well. However, the resulting image lacks high-frequency details of the original style exemplar, contains artifacts, and colors that are not present in the original style. The right column represents our method where low-resolution neural transfer result is used as a guidance channel for patch-based style transfer. Our result attenuates the neural artifacts and restores the original color and texture of the style exemplar.*

computational overhead instead of running patch-based synthesis on the full resolution, we only upscale the dense correspondence field computed at a lower resolution level. We demonstrate that such a simple upscaling step can be performed quickly while still providing comparable visual quality as the full-fledged synthesis. This enables us to achieve high-quality stylization of extremely large images (see Fig. 4.1 where an image of 346Mpix is stylized). Our approach is generalized and can utilize any existing neural stylization method. We demonstrate this generality on a variant of our style transfer algorithm that directly uses the response of a neural network as a guide for patch-based synthesis. We developed a prototype of our method in the form of a Photoshop plug-in and put it into the hands of professional artists.

4.2 Related Work

One of the key tasks of non-photorealistic rendering [Kyp+13] is to deliver stylized depictions of photos or synthetic scenes which preserve high-level information captured in the scene while on a detail level the resulting image resembles the artistic look.

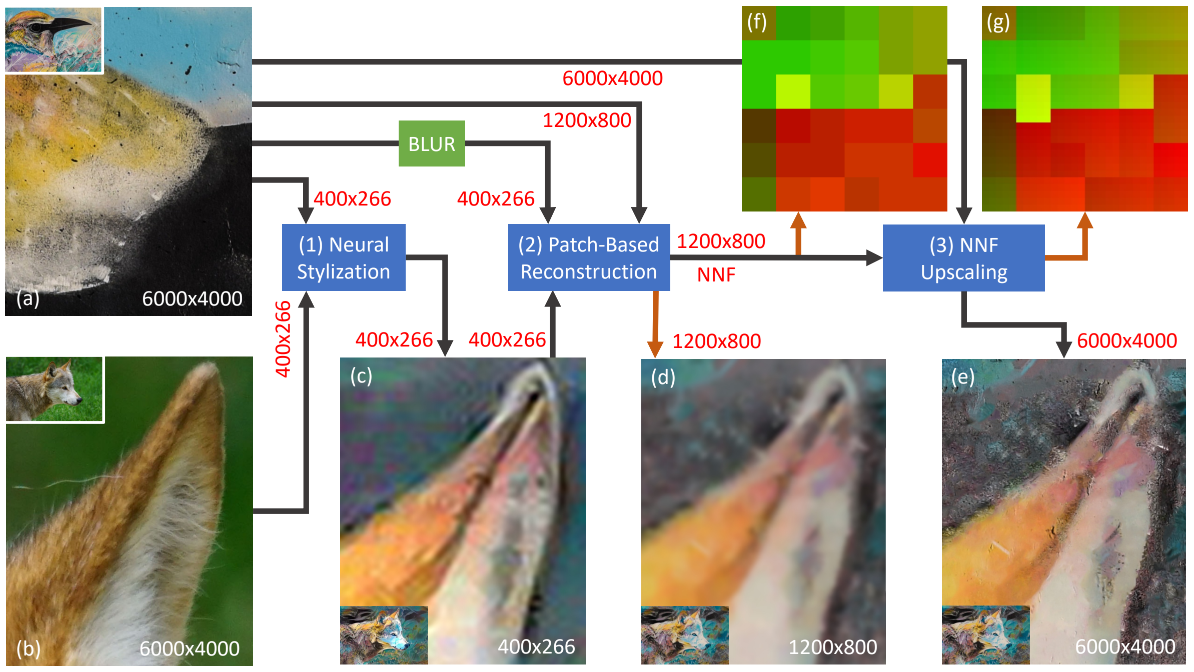


Figure 4.4: *Proposed pipeline: (a) style exemplar and (b) content image are both subsampled α–times and processed by a neural-based style transfer method (Sec. 4.3.1) which results in low resolution image (c) where fine details are missing and artifacts are apparent (see green and purple checkerboard artifacts). Next, low resolution result (c) from the previous step, style image (a) in the same resolution as (c), and β–times subsampled style image (a) are used as an input to a patch-based synthesis algorithm (Sec. 4.3.2) which outputs dense nearest neighbor field (NNF) (f) from which the corresponding image (d) can be produced using voting step [WSI07]. Finally, in NNF upscaling step (Sec. 4.3.3) the low-resolution NNF (f) is upscaled β–times to the original resolution (g). Patch coordinates in NNF (f) and (g) are encoded as red and green color levels. Note subtle color gradients in (f), which indicate the presence of fine patch coordinates in upscaled NNF that points to the patches in the original high-resolution style exemplar (a). Given the upscaled NNF (g) and the style exemplar in its original resolution (a), high-resolution, and a perfectly sharp final result is created using voting step (e).*

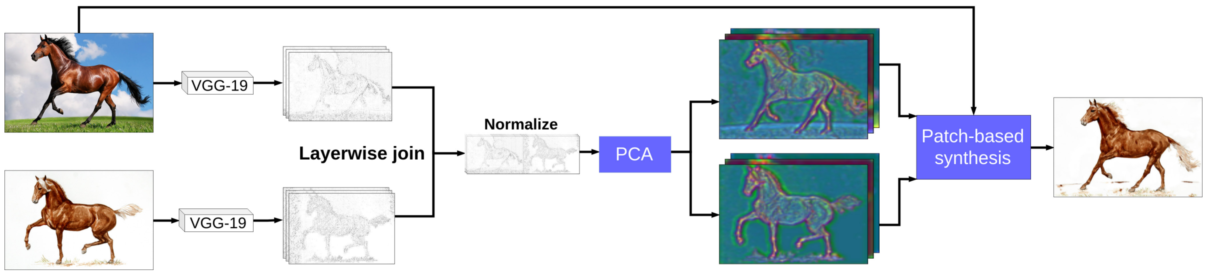


Figure 4.5: *An overview of our VGG-guided style transfer pipeline: we start with a target image and a style exemplar, extract their VGG-19 features, normalize them, reduce their dimensionality using PCA, and use these as guidance for subsequent patch-based synthesis. Even though the proposed pipeline is straightforward, it yields convincing output.*

Stroke-based approaches were one of the first techniques that enabled generation of stylized imagery. Rotated and translated brush strokes from a predefined set are placed according to some guiding information (e.g., the direction of image gradients). This technique is applicable both in 2D [Her98] and 3D [Sch+11] environment producing quite compelling results. Nevertheless, the main drawback here is the restriction to a predefined set of strokes, which limit the variety and fidelity of the stylized output. Such a limitation can partly be alleviated by introducing example-based brushes [Lu+13; Zhe+17]; nevertheless, the final look is still limited to a subset of styles that can be simulated by a composition of brush strokes.

To address this issue a more robust and general example-based approach called *Image Analogies* was pioneered by Hertzmann et al. [Her+01]. Given an arbitrary style exemplar and a set of guidance channels, the stylized image can be produced using guided patch-based synthesis [WSI07; Kas+15; Fi16]. This approach has been successfully applied to various stylization scenarios including fluid animations [Jam+15], 3D renders [Fi16; Sýk+19], facial animations [Fi17] or video clips [Jam+19]. Nevertheless, a common drawback of this method is that it requires the preparation of custom-tailored guidance to deliver compelling stylization quality. Furthermore, an extensive computational overhead at higher resolutions makes those techniques difficult to use in production.

Neural-based style transfer approaches recently became popular due to advances made by Gatys et al. [GEB16], they successfully applied the pre-trained convolutional neural network VGG [SZ14] to the problem of style transfer. The core idea of their method is to match statistics in the domain of VGG [SZ14] features of both the content and style images. They further extended this idea in [Gat+17] to introduce control over spatial location, color information, and scale of features. While these techniques produce impressive results for some particular style exemplars, they usually suffer from loss of high-frequency details of the style exemplar which is inevitably caused by the convolutional nature of the underlying neural network. Moreover, mentioned neural techniques usually have considerable computational overhead and memory footprint.

Although a feed-forward network can be pre-trained to speed up the stylization [JAFF16; Uly+16a; DSK16; Che+17], every new style requires additional costly training. Recently, adoption of encoder–decoder scheme was proposed [Li+17; HB17; Lu+17] to enable arbitrary style transfer in a feed-forward fashion. Here the encoder, usually convolution layers of the VGG, is used to get the feature representations (statistics) of the content and style, which are then combined, and a pre-trained decoder is used to turn the latent features back into the image. Nevertheless, all these techniques still suffer from convolutional artifacts leading to a lower quality of the synthesized imagery at a pixel level.

Recently, attempts to combine patch-based and neural-based techniques were proposed. Li et al. [LW16a] search local neural patches from the style image concerning the structure of a content image, which leads to better reproduction of local textures. Liao et al. [Lia+17] later extended this idea in their *Deep Image Analogy* framework which adapts the concept of *Image Analogies* [Her+01] in the domain of VGG features. Gu et al. [Gu+18] recently proposed to perform reshuffle in spirit of [Kas+15] to reduce the overuse of particular features. Futschik et al. [Fut+19] use patch-based method [Fi17] to generate a larger dataset of stylized portraits which is then used to train a generative adversarial network capable of reproducing similar quality results as those in the underlying dataset. Although these techniques can notably improve the stylization quality and better preserve high-frequency details, they still heavily rely on the space of VGG features and do not explicitly enforce textural coherence on a pixel level in color domain [WSI07] which is essential to retain the fidelity of the original style exemplar.

4.3 Our Approach

We propose an approach to combine patch-based synthesis with neural style transfer methods. The proposed pipeline overcomes three crucial obstacles which prevent existing stylization approaches from being used in real production: first, lower texture quality of neural-based techniques; second, the necessity of specific guidance for patch-based methods; and third, the resolution limitation which affects the usability of both approaches. Our framework allows easy switching to the newest future inventions in either neural-based or patch-based techniques.

As our first step, given the exemplar *Style* and the target image *Content*, we use an arbitrary neural-based style transfer method to synthesize an initial result (see Fig. 4.3 middle column). The resulting image on its own lacks high-frequency details of the style exemplar, and contains artifacts such as geometric distortions and colors that are not present in the original style. Also, the original contrast is usually artificially exaggerated, and edges are not sharp. However, on the other hand, it nicely preserves global style properties such as color distribution and respects the image semantics in general.

Our key idea is to use the low-resolution neural style transfer result as a guiding channel for patch-based synthesis. This enables us to combine the advantages of both techniques and to address the aforementioned limitations (see Fig. 4.3 right column). In particular, a pair of guidance channels *Source* and *Target* is needed for guided patch-based synthesis. We use blurred style exemplar as the *Source* guide and the low-resolution neural style transfer result as the *Target* guide. After running the guided patch-based synthesis, our result (Fig. 4.3 right column, bottom) effectively attenuates the neural artifacts and restores the color and texture of the original style exemplar.

Fig. 4.4 illustrates our entire pipeline which consists of three main parts: neural-based style transfer method, guided patch-based synthesis, and nearest neighbor field (NNF) upscaling method. Those individual steps are described in more detail in the following sections.

4.3.1 Neural-Based Style Transfer

Both *Style* (Fig. 4.4a) and *Content* (Fig. 4.4b) images are first subsampled by a coefficient α. This step is necessary not only to overcome the resolution restrictions but, more importantly, to suppress various high-frequency artifacts caused by neural-based techniques (α essentially defines the *working resolution* of a neural-based method). The α–times subsampled neural-based result (Fig. 4.4c) is then used as a guide for the patch-based synthesis method. Its resolution will be improved later in our pipeline.

4.3.2 Guided Patch-Based Synthesis

The output from the neural method (Fig. 4.4c) is used as a *Target* guide image in the patch-based method. Our pipeline does not assume any particular patch-based method; we used StyLit [Fi16] algorithm for synthesis, however, we adapt its original error metric for measuring patch similarity to our needs. Let $\mathcal{S}$ be a style exemplar, $\mathcal{O}$ an output image, and $G^{\mathcal{S}}$ and $G^{\mathcal{T}}$ source and target guides, for matching two patches $p \in G^{\mathcal{S}}$ and

$q \in G^{\mathcal{T}}$; we use the following error metric:

$$E(\mathcal{S}, \mathcal{O}, G^{\mathcal{S}}, G^{\mathcal{T}}, p, q) = \\ ||\mathcal{S}(p) - \mathcal{O}(q)||^2 + \lambda_g ||G^{\mathcal{S}}(p) - G^{\mathcal{T}}(q)||^2 \quad (4.1)$$

where λ_g is a weighting factor for guiding channel and the first term helps to preserve *texture coherence* by directly matching colors in patches of *Style* to those in the output image $\mathcal{O}$. Of all the images, only $\mathcal{O}$ is iteratively updated during the optimization process described in StyLit [Fi16].

To obtain *Source* guide image, we use the already subsampled style image used in the previous step (Sec. 4.3.1), and upsample it back to its original resolution. To encourage the patch-based synthesis to find good correspondences for the style transfer, equivalent subsampling followed by upsampling needs to be done for both the *Source* and *Target* images. In spirit of *Color Me Noisy* [Fi14], an additional low-pass filter can be applied on the *Source* image to let the synthesis algorithm deviate more from the initial solution, thus making the final result more abstract.

In Fig. 4.4d the result of patch-based synthesis is depicted in color for clarity, nevertheless, internally in our processing pipeline we use only the resulting nearest neighbor field (Fig. 4.4f) which is subsequently upsampled (Fig. 4.4g) and turned into a high-resolution image in the next step.

4.3.3 NNF Upscaling

Given the computed NNF–nearest neighbor field (Fig. 4.4f) and the style exemplar in its original resolution (Fig. 4.4a), a *voting step* (c.f. [WSI07]) needs to be performed in order to reconstruct the final image. To reduce the computational overhead, we perform the patch-based synthesis (Sec. 4.3.2) at β–times lower resolution than the original target resolution (thus β essentially defines the *working resolution* of a patch-based method). Next, the resulting **nnf** (Fig. 4.4f) is upscaled by a factor of β to obtain the **NNF** (Fig. 4.4g) of the same resolution as the target image as follows:

$$\mathbf{NNF}(x, y) = \mathbf{nnf}(x/\beta, y/\beta) \cdot \beta + (x \bmod \beta, y \bmod \beta) \quad (4.2)$$

Finally, we perform a voting step using **NNF** to produce the final high-resolution result precisely preserving the characteristics of the canvas and the original artistic medium (Fig. 4.4e).

4.4 VGG-Based Guidance

One of the limitations of the proposed base algorithm introduced in the previous section is that it relies on color information to establish correspondences between style exemplar and the target image. This drawback could lead to an ambiguity that may introduce visible stylization artifacts (see Fig. 4.6).

In this section, we introduce a variant of our style transfer pipeline that uses features extracted by the convolutional layers of a classification network for guidance directly rather than relying on a neural style transfer algorithm to produce initial color domain stylization. The aforementioned neural responses provide more discriminative guidance

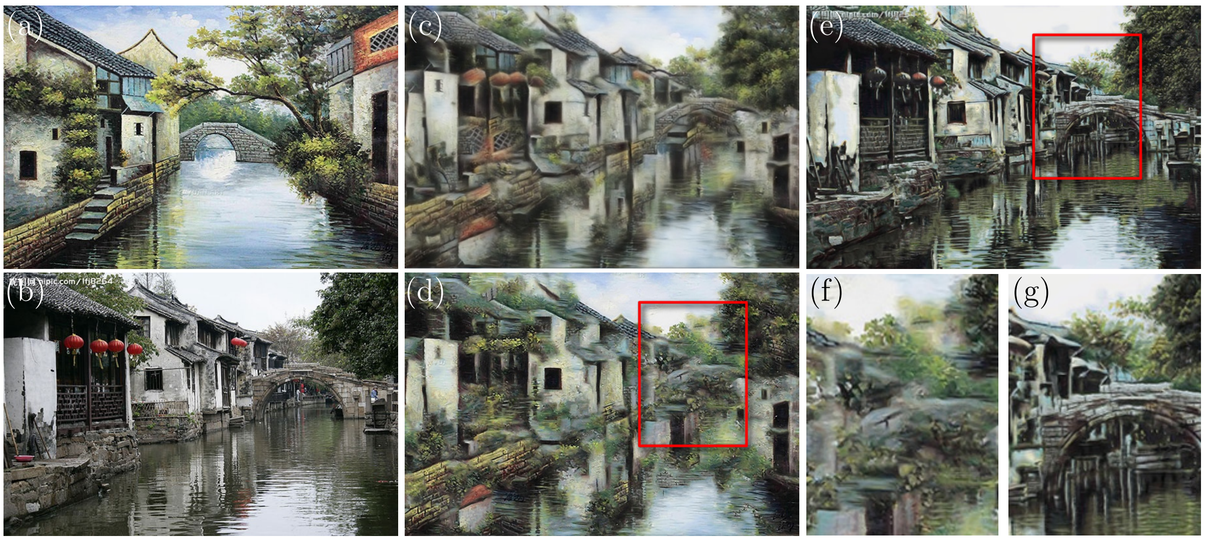


Figure 4.6: *Demonstration of the problem when patch-based synthesis has to rely on ambiguous color guidance: (a) style exemplar, (b) target image, (c) output of Gu et al. [Gu+18], (d) output of our basic algorithm with color-based guidance, (e) output of our style transfer algorithm with neural guidance. Note how our VGG-guided algorithm better preserves the semantics of the target photo, cf. details in (f) and (g).*

than colors and thus can preserve global semantics of the target while still keeping the benefits of patch-based optimization.

Our approach is inspired by modern optimization-based neural style transfer techniques of Liao et al. [Lia+17] and Gu et al. [Gu+18] that rely on computationally demanding global descent through a complicated loss function using an optimizer like L-BFGS. Although this approach is conceptually similar to the patch-based optimization framework, in our case expensive global descent is approximated by a highly efficient approximate nearest-neighbor matching.

The algorithm first extracts neural features for both the source and target image in multiple scales (see Fig. 4.5). Specifically, we run the input images through the neural network on four resolutions: 1344×1344, 896×896, 448×448 and 224×224. This set was chosen to capture a broader range of neural features.

For this purpose, we use VGG-19 network architecture trained on the ImageNet dataset [SZ14]. After running a feed-forward pass on the input image, features are extracted from 6 different layers of the network. The layers used are $conv2\_2$, $conv3\_1$, $conv3\_4$, $conv4\_1$, $conv4\_4$, and $conv5\_1$. Features are extracted after applying the ReLU activation.

These neural features capture localized semantic similarities found in both images and can be used to guide the patch-based synthesis. However, the high dimensionality of these per-pixel features might significantly compromise both the performance and the quality of the patch-matching step. To avoid this, we reduce the feature dimension using PCA [TP91]. In particular, we treat each feature vector as an independent point and process feature maps in groups of the same resolution. The number of principal components we extract varies by feature map resolution. We use top 3 components at 1344×1344, top 6 components at 896×896, and finally top 12 components for the two remaining resolutions. We normalize the resulting values to $[0, 255]$ interval and

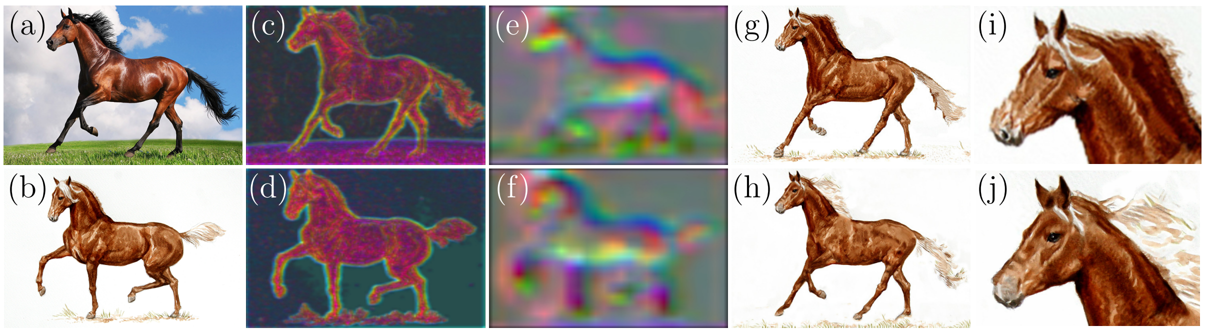


Figure 4.7: *An example result from our VGG-guided style transfer algorithm: (a) target image, (b) style exemplar, corresponding compressed VGG-responses of low- (c, d) and high-level (e, f) features used as a guide for patch-based synthesis, (g) output of Liao et al. [Lia+17], (h) output of our style transfer framework with neural guidance, note how our method can deliver comparable visual quality, cf. details in (i) and (j).*

resample them to the required resolution using bicubic upsampling. This can either be lower resolution, typically used in neural techniques, or full resolution of the target image. Lastly, we run the patch-based synthesis algorithm of Fišer et al. [Fi16] to produce the final stylized image. The output is visually comparable to the state-of-the-art [Lia+17; Gu+18] (see Fig. 4.7).

4.5 Results

We implemented our method both for CPU and GPU, using C++ and CUDA, respectively.

The parameter α is set to make the input images to the neural-based method approximately 400–500 pixels wide. In the case when the input images are already of low-resolution, we set α to be at least 2—to ensure the patch-based synthesis will have enough freedom to fix some of the artifacts caused by the neural-based approach. The α—sub-sampling allows us to get the result from a neural-based approach much faster or use a method that does not support high-resolution input. Moreover, it allows us to significantly suppress some of the artifacts of neural approaches. The parameter β allows us to stylize images of size 346Mpix or even larger, and to get the final result much faster (see an extreme-resolution result in Fig. 4.1 and our supplementary material). We observed that if the parameter β is in range 1–4, the perceived loss in the quality is almost negligible. If the parameter β is in range 6–10, when zooming closely, one can observe some repetition artifacts, however, the image is sharp and the overall quality is still satisfactory.

We measured run-time and memory performance. For detailed run-time measurement on mid-range laptop see graph in Fig. 4.8. On a desktop PC, the computational overhead is even lower, e.g., on NVIDIA Quadro M2000, stylizing the image of size 160Mpix takes between 3–30 seconds depending on the selection of the parameter β. Increasing the parameter β causes a linear increase in the computational time, while the number of pixels grows exponentially. Our method requires a few hundred MBs of RAM/GPU memory. The exact amount depends on the resolution of the input images and the value of the parameter β.

The performance of the neural-based step depends on a particular method. However, because the input is of very low resolution, 400–500 px wide, the run-time typically ranges between hundreds of milliseconds and several seconds. Most neural-based approaches cannot stylize images larger than 4K-by-4K due to GPU memory constraints. Although there is a possibility to decompose the synthesis into a set of tiles that are processed separately and stitched together, the resulting image would still suffer from the convolutional nature of used neural network introducing disturbing high-frequency artifacts and colors not present in the original style exemplar.

We plugged several different state-of-the-art neural-based style transfer techniques into our framework (see Fig. 4.9 and 4.10). In all cases, applying patch-based synthesis with neural transfer output as guidance produces better results than using the neural-based approach alone. The most noticeable differences are visible in (1) the original colors (e.g., saturated pixels that do not appear in the original style exemplar are removed), (2) suppression of checkerboard artifacts caused by deconvolution [ODO16], and (3) results are sharper containing important high-frequency details of the original brush strokes and underlying canvas structure. Fig. 4.1 demonstrates stylization of a 346Mpix image. Despite the huge resolution, the result is still perfectly sharp and preserves well characteristics of the original artistic media.

To demonstrate the benefit of using the output of the neural approach to guide the patch-based synthesis, we compared our method to the guidance based only on blurred grayscale images (Fig. 4.3 left column) as proposed in the original Image Analogies method [Her+01], the result does not properly respect the content semantics, causing trees to become pink.

In Fig. 4.11 and 4.12, we present additional results of our VGG-guided style transfer algorithm. These demonstrate the proposed method can produce convincing stylization without the need to use existing neural techniques as a preprocess.

Finally, in Fig. 4.13, we demonstrate a UI prototype of our method running in Photoshop.

4.6 Limitations and Future Work

Although in most cases, our approach is capable of delivering significantly better and visually more pleasing results than the underlying neural technique itself, it still relies on the neural result as the initial solution. Due to this reason, we cannot fix large-scale artifacts produced by the neural-based method (see Fig. 4.14). In the current pipeline, only high-frequency artifacts can be suppressed. When zooming in, the improvement in the texture quality is immediately visible, nevertheless, looking from a distance, high-resolution image obtained by our method may appear almost identical as the result of the underlying neural approach.

As future work, we would like to tackle the issue commonly seen in neural techniques, i.e., many different colors are mixed together within a single coherent region or when the same mixture of colors is used to stylize semantically different regions (see an example in Fig. 4.15). To address this problem, we see two promising solutions. First, extending our pipeline in a way that patch-based synthesis is guided by a neural network trained for segmentation on both natural and artistic images to encourage more semantically correct matching of patches. Second, incorporate mask-based loss function as described

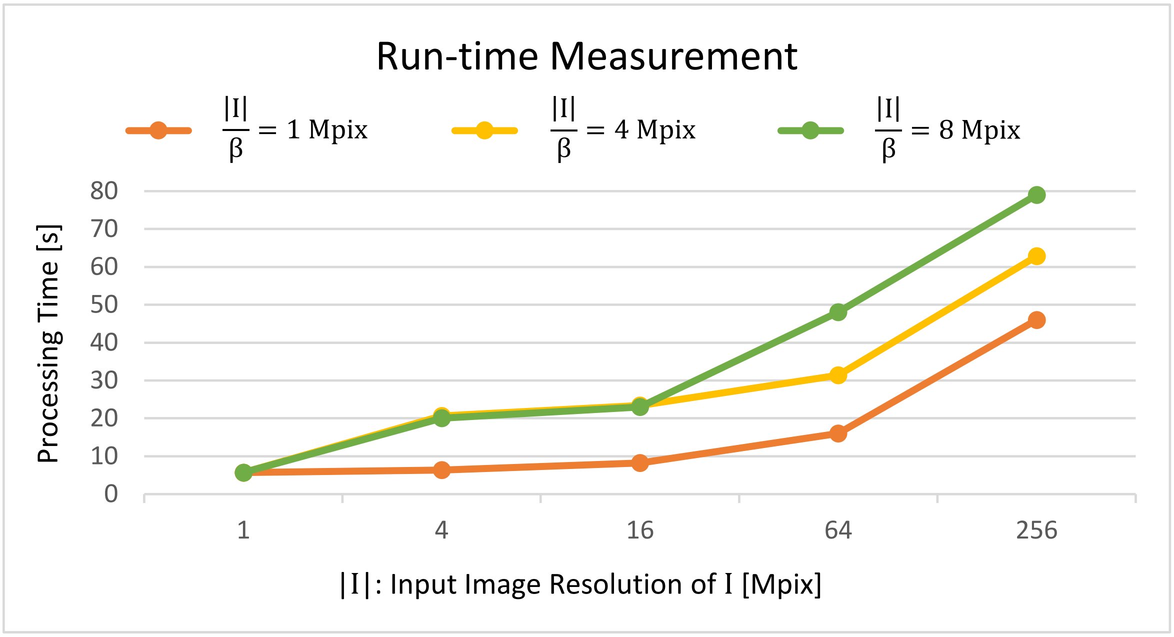


Figure 4.8: *Performance of our method (full pipeline–Fig. 4.4, excluding the neural part) on images ranging from resolution of 1Mpx, (i.e.* 1000 × 1000 *px) to extremely large resolution of 256Mpix (i.e.,* 16000 × 16000 *px). Orange, yellow, and green lines show a case where the parameter* β *was set such that the patch-based method was run on a resolution of 1Mpix, 4Mpix, and 8Mpix respectively. The measurement was done on a mid-range laptop with NVIDIA GTX 1050 graphics card.*

in [Rei+19]. Although, this might not be feasible for all neural-network approaches we use or in a case when it is desired to treat an underlying neural-network as a black box.

Our technique helps to restore high-frequency details and essential attributes of used artistic media; however, in some cases, this process might destroy some of the important content details. We see a promising solution in the work of Calvo [Cal+19], where they introduce a technique to intensify or reduce the stylization strength locally.

Another interesting follow-up of our work could be an extension to videos. This might seems straightforward, but even if the video delivered by the underlying neural-based style transfer method is stable in time, randomness in the patch-based step of our pipeline will most likely introduce disturbing temporal inconsistency. To solve this, one could use techniques described in [Jam+19] or [Fi17].

Another area for future work worth exploring would be adding interactions to control the result. Also, some of the neural-based approaches support multiple style exemplars; we suggest to explore possibilities of using multiple styles in our enhancing scenario.

4.7 Conclusion

We have presented a new approach that combines neural and patch-based style transfer techniques, and proposed a way to utilize the generality of the former, while achieving the texture quality of the latter. We introduced a computationally inexpensive algorithm for upscaling the synthesis output to obtain its high-resolution version and a new approach to

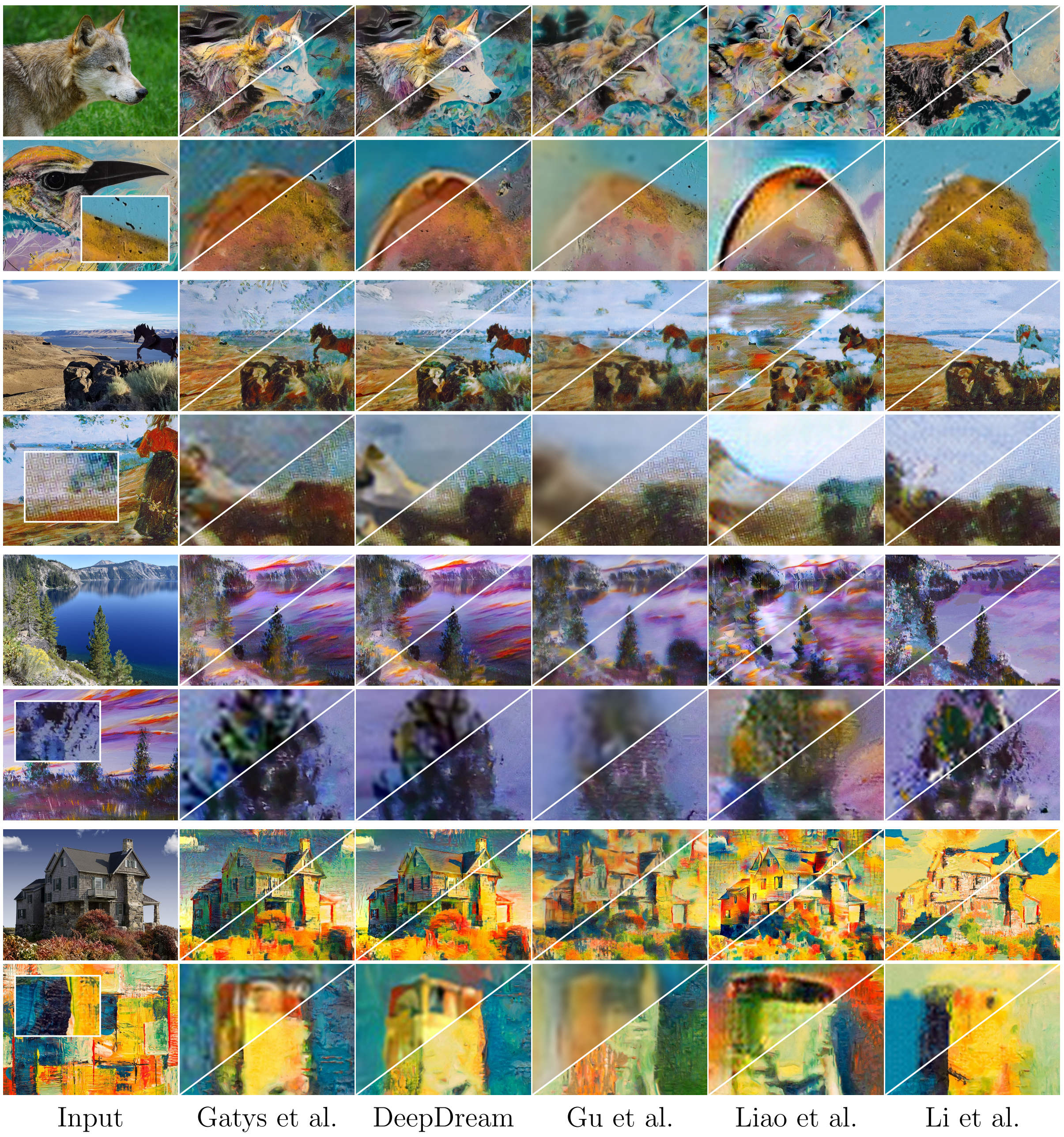


Figure 4.9: *Our method enhancing the results of five different neural-based approaches: The leftmost column–content images and style exemplars (with zoomed patches). Next, left-to-right, are the result of DeepArt [GEB16], DeepDream, Gu et al. [Gu+18], Liao et al. [Lia+17], and Li et al. [Li+17]. The top-left triangle shows the result of the underlying neural-based approach (bicubically up-sampled from a typical size of* 600×400 *px to the target resolution), while the bottom-right shows result enhanced by our method (top row–entire stylized images, bottom row–zoom-in). Our results not only have significantly higher resolution but also better preserve the original colors and canvas structure as well as brush strokes visible in the exemplar painting. Various artifacts caused by the neural approach are significantly suppressed. All images shown in this figure are of resolution ranging from* 4000×2200 *to* 6000×4000 *px.*

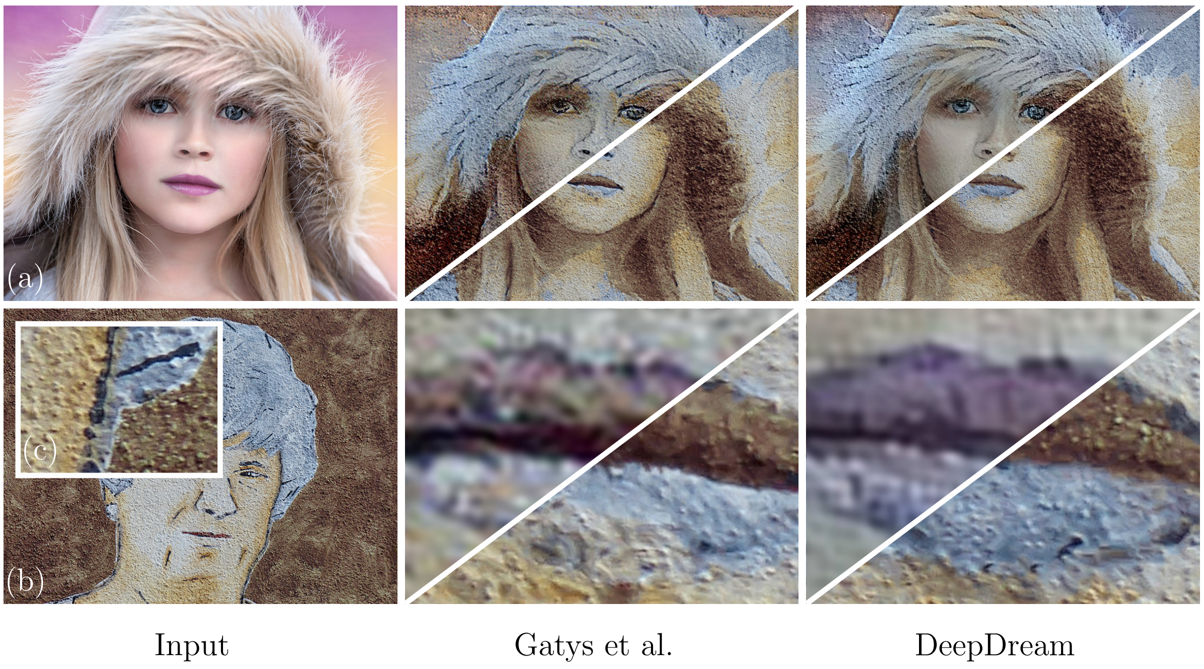


Figure 4.10: *Portrait on a wall: (a) target content of resolution* 4000 × 3000 *px, (b) style exemplar of a painting on a wall having the same resolution, (c) 10x zoom-in to the (b) to show fine artistic attributes and structure of the canvas–wall/plaster. Our method is entirely independent of the used artistic medium as well of a canvas the style exemplar is presented on. The results are presented in the same fashion as in Fig. 4.9.*

neural-based style transfer that can use responses of the neural network directly as a guide for patch-based synthesis. Thanks to those advances, we can produce style transfer results with notably larger resolutions than previous neural-based techniques and significantly reduce the computational overhead while retaining comparable visual quality. We believe our method could enable broader applicability of style transfer methods in commercial practice. To that end, we integrated our approach into Adobe Photoshop in the form of a plug-in.

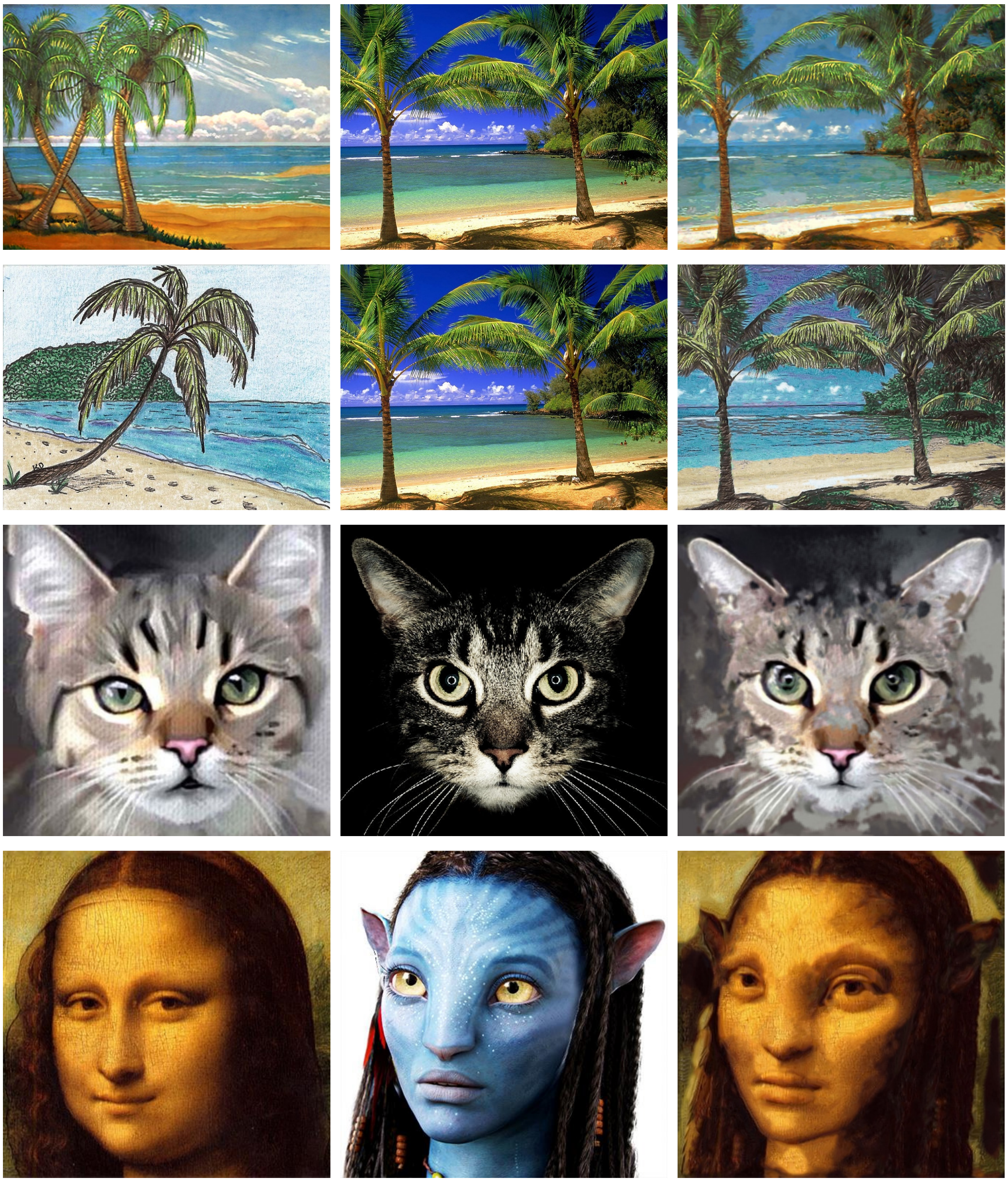


Figure 4.11: *Results produced by our VGG-guided style transfer algorithm (from left to right): style exemplar, target image, and our result. Our method works well namely in cases when style and target images depict similar content, i.e., when they have compatible VGG activations.*

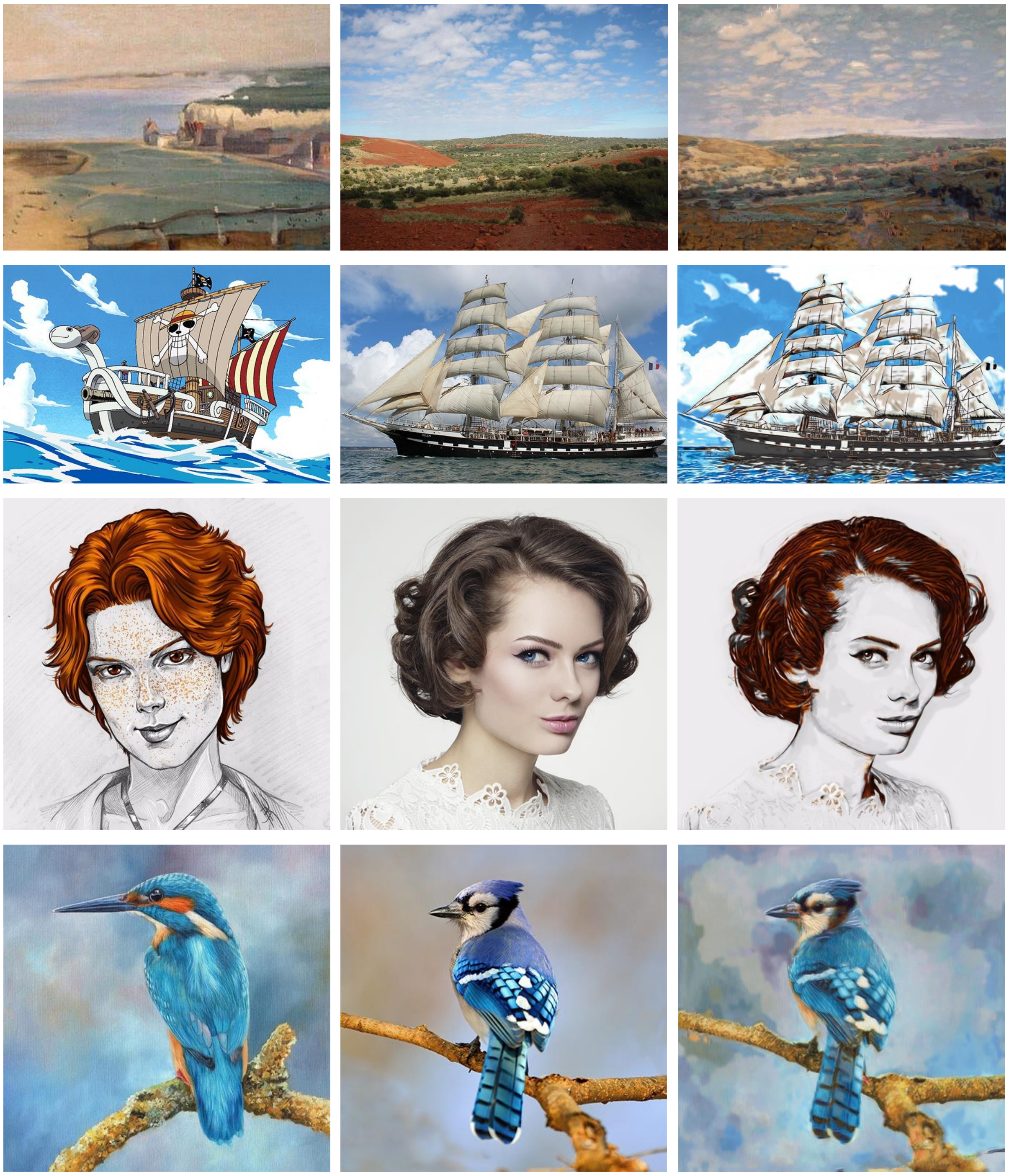
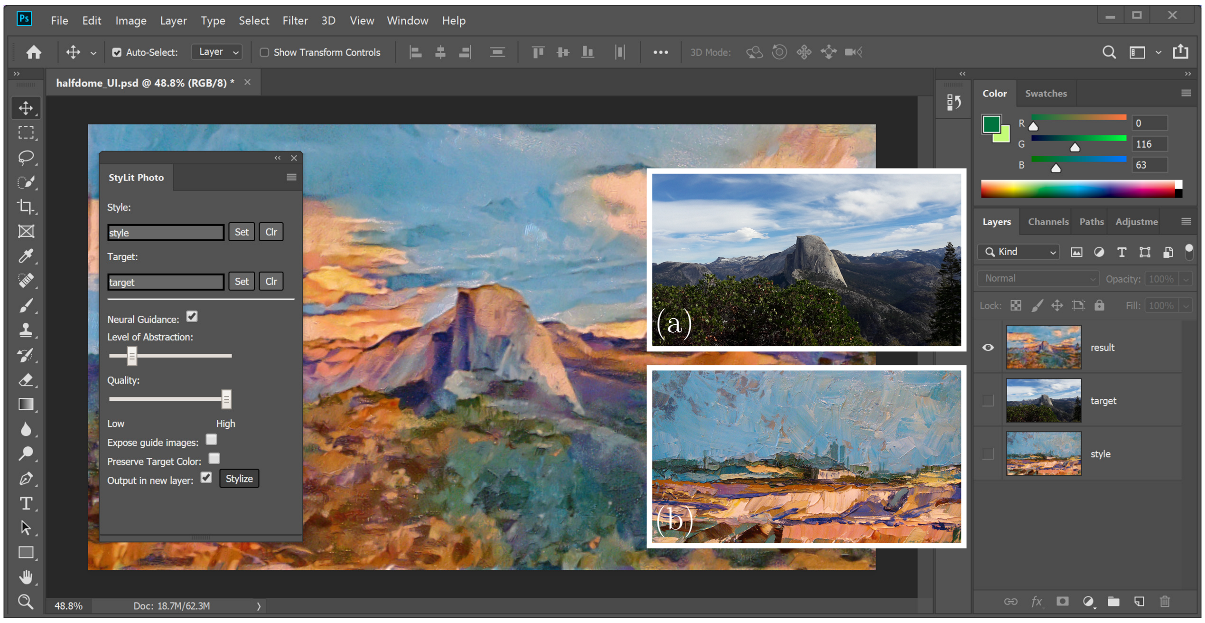


Figure 4.12: *Additional results produced by our VGG-guided style transfer algorithm (from left to right): style exemplar, target image, and our result.*



StyLit Photo
Style: style [Set] [Clr]
Target: target [Set] [Clr]
Neural Guidance: ✓
Level of Abstraction:
Quality: Low High
Expose guide images:
Preserve Target Color:
Output in new layer: ✓ [Stylize]

R 0
G 116
B 63

Figure 4.13: *A screenshot of our method running in Adobe Photoshop: (a) zoom of a target layer, (b) zoom of a style layer; the visible layer is the result of DeepDream enhanced by our method.*

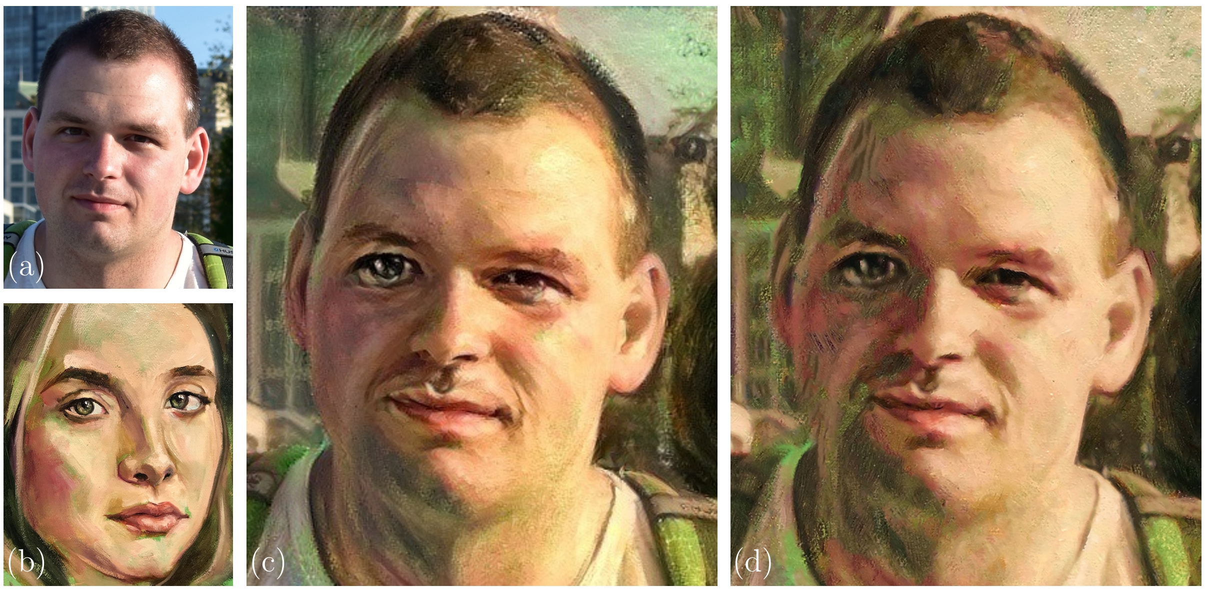


Figure 4.14: *Large-scale artifact limitation: (a) content image, (b) style exemplar, (c) result of Gatys et al., distortions in eye region are visible, (d) ours, colors and high-frequency details are reproduced well; however, in our current pipeline, large-scale artifacts produced by the underlying neural approach are not fixed. Thus distortion in the eye region is still apparent.*

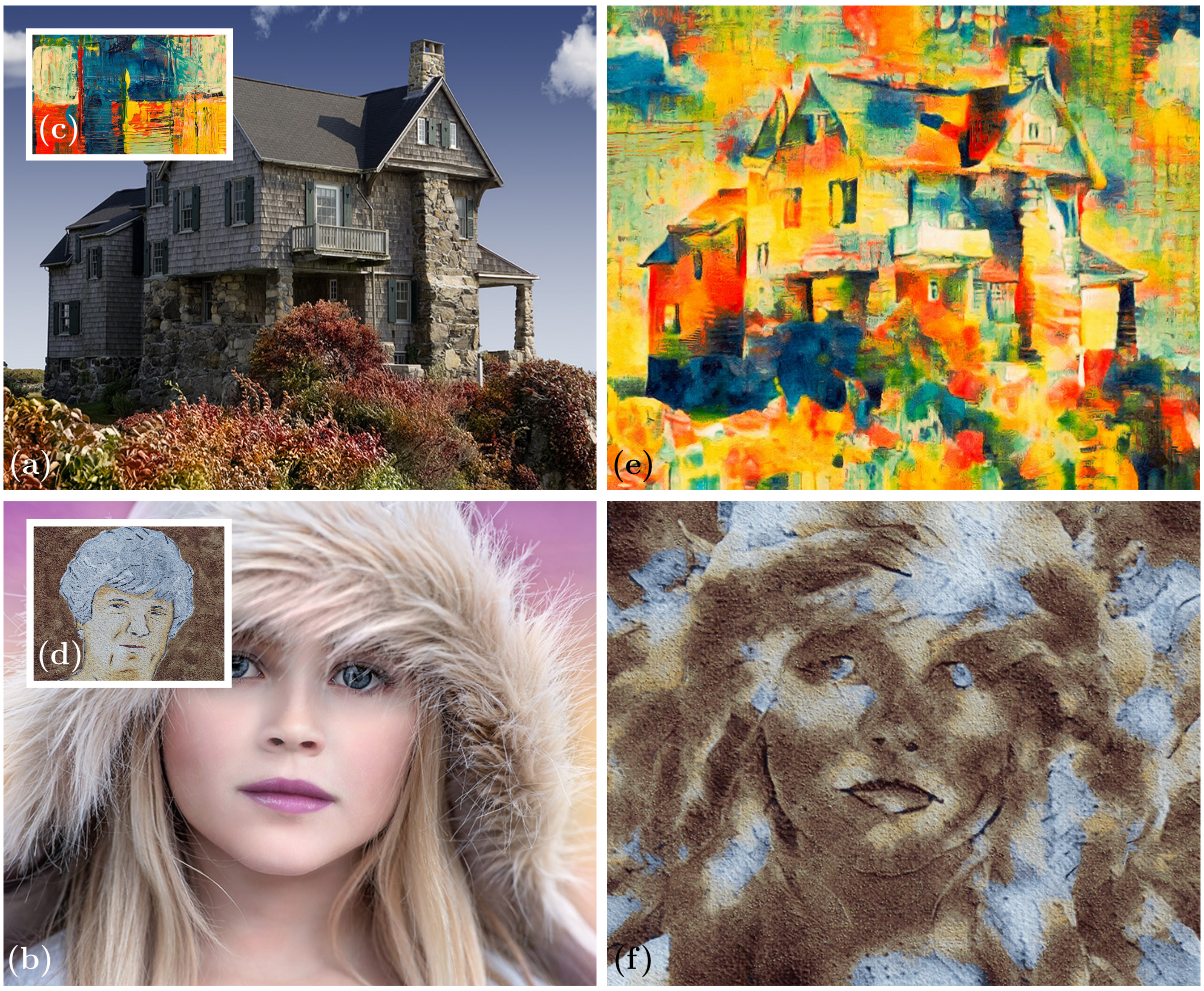


Figure 4.15: *A limitation common to neural-based approaches: (a-b) content image, (c-d) style exemplar, (e-f) result of [Li+17] enhanced by our method. The content of the original image is not preserved well. In the first case, the similar mixture of colors is used to paint bushes, house, and also the sky. In the second case, all colors appearing in the style exemplar are used to stylize the target regardless of its content. However, high-frequency content is reproduced well. To address this limitation, we propose to incorporate a neural network trained for image segmentation into our pipeline.*

Chapter 5

Stylizing Video by Example

5.1 Introduction

In the past decades, advances in computer graphics led to a revolution in the art of animation, giving birth to an entirely new branch of animation which is three-dimensional, and includes photorealistic lighting effects and physically accurate simulation. Together with lighting, material, and performance capture, the production pipelines of animated video now resemble live-action production more closely than traditional animation. An unfortunate side effect of this is that, due to production and technical considerations, there is a "style gap" between traditional and 3D animation, where the latter has its own distinct look, and it has so far been impossible to convincingly reproduce the look of the former using the aforementioned production pipelines. Currently, there are no automated methods that could use live-action performance capture to produce the look of traditional animation. Although artists have attempted to bridge this gap for, e.g., abstract stylization (*A Scanner Darkly*[1]) or painterly look (*Loving Vincent*[2]), these were monumentally laborious efforts that had to be created manually frame-by-frame.

One possible way to overcome this could be to employ example-based style transfer techniques to transfer artistic style from a traditionally created style exemplar to a synthetic or live action target. This approach recently became popular thanks to advances in neural [GEB16; RDB18] and patch-based transfer [Fri+16; Fri+19] techniques. These approaches can alter the global appearance of the target to roughly resemble the given visual style, but the effectiveness of stylization relies solely on the internal representation of style and content of the respective algorithm. They do not offer users any explicit controls and cannot fulfill the need of artists to precisely express their artistic intent.

In an effort to provide this sort of control over style transfer, Hertzmann at el. [Her+01] pioneered an Image Analogies framework where the style exemplar, as well as the target, are extended with additional guiding channels which provide spatial control of how the style is transferred. This ensures that particular features of the style exemplar will appear at desired locations in the target. It was shown recently [Bén+13; Fi16] that this additional control allows for a more semantically meaningful transfer and results in higher visual quality than the generic methods [GEB16; Fri+19].

[1]https://en.wikipedia.org/wiki/A_Scanner_Darkly_(film)
[2]http://lovingvincent.com

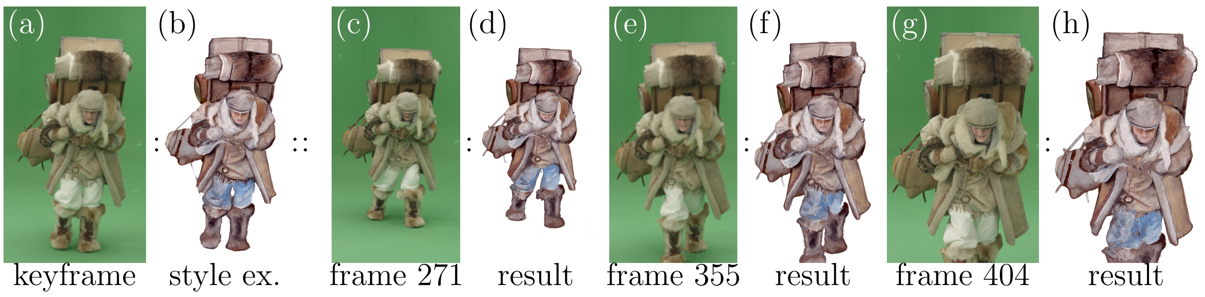


Figure 5.1: *An example of a stylized sequence produced by our approach. One frame from the sequence is selected as a keyframe (a) and a corresponding style exemplar is painted using watercolor (b). Then, for the rest of the sequence (c, e, g) our technique produces stylized output (d, f, h) which preserves the artistic attributes of the specified style exemplar, reflects structural changes in the target video, and maintains temporal coherence. Video frames (a, c, e, g) courtesy of © MAUR film, style exemplar (b) courtesy of © Pavla Sýkorová, used with permission.*

The main drawback of this approach is that the guidance channels need to be generated first. Much research was done into algorithmic solutions for specific scenarios (e.g., rendering attributes from known geometry [Bén+13; Fi16] or using landmark detectors and face segmentation [Fi17]), but guidance generation for the general case of arbitrary images remains an open problem. Efforts into neural-based guidance by Liao et al. [Lia+17] and later Gu et al. [Gu+18] demonstrated that the response of VGG net – a deep neural network trained for object classification [SZ14] – can be used as a guide to automatically control the transfer in certain cases. Unfortunately, this approach is able to reliably discriminate features only on the type of images VGG was trained on (faces, animals, objects, etc.). In a more general scenario, the accuracy is insufficient and may lead to obvious inconsistencies (see, e.g., transfer of facial patterns to the legs of the target subject in Fig. 7.7). Moreover, those techniques do not address temporal coherence which is crucial for video synthesis.

In this paper, we formulate an alternative analogy-based approach for the artistically controlled stylization of video, which (a) addresses the style gap by facilitating free-form artistic stylization of synthetic or live-action video sequences, (b) gives artists control by allowing them to explicitly specify local styles using traditional painting techniques that are familiar to them, and (c) does not require "insider knowledge" of the target content, such as segmentation, landmarks, 3D or rendering information.

We build on the keyframe stylization paradigm proposed by Benard et al. [Bén+13], where the artist paints one or more keyframes in a preferred style, and the algorithm then propagates the specified style to the rest of the sequence. Our key difference from the aforementioned approach is that we do not require any knowledge of the underlying 3D structure of the target scene. Instead, we obtain semantically meaningful transfer by using the original color information from the input video together with approximate positional and temporal guidance generated using optical flow estimation. We further show that when used in conjunction with a state-of-art patch-based synthesis algorithm [Fi16], this guidance results in superior visual quality while preserving the artistic intent. Finally, we demonstrate the practical utility of the proposed approach on examples from real production settings.

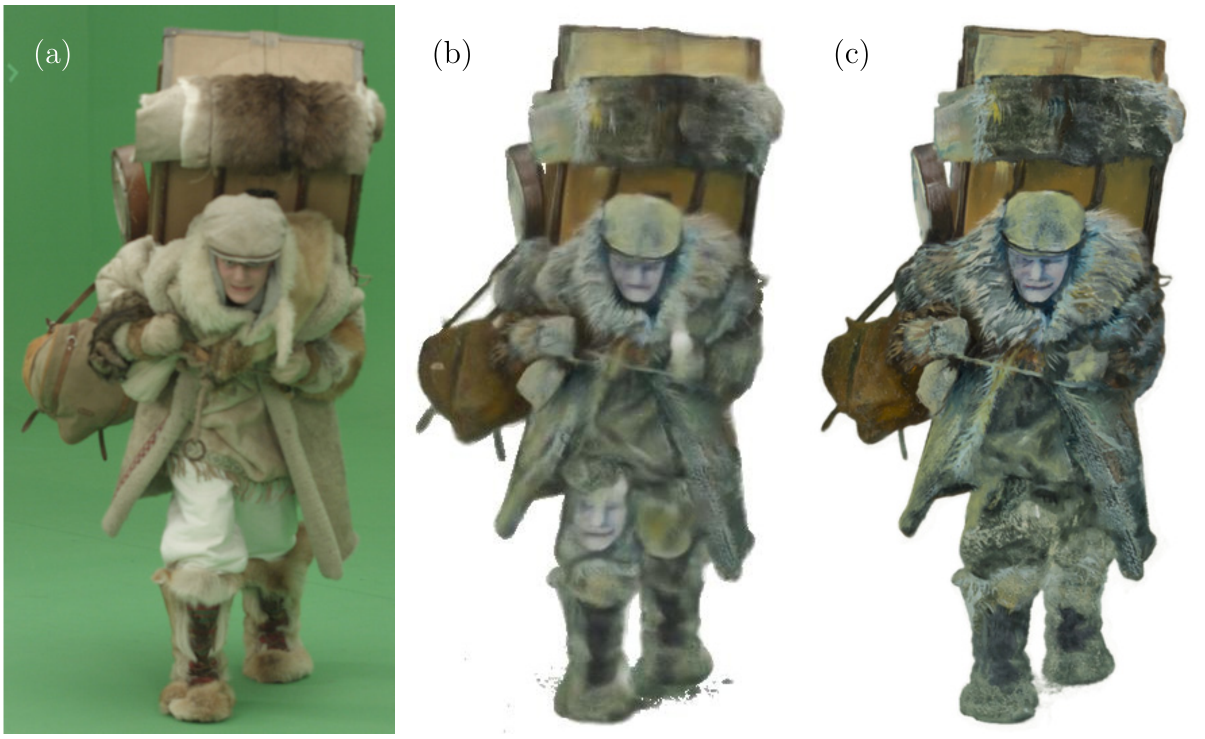


Figure 5.2: *Common neural stylization artifacts. Style transfer guided by the response of VGG network might transfer the facial pattern onto the leg region. (a) target frame, (b) result of Gu et al. [Gu+18], (c) our approach. Video frame (a) courtesy of © MAUR film, used with permission.*

5.2 Related Work

Traditional image and video stylization methods employ algorithmic filters hand-crafted to transform an input image or video to a particular style. These can be based on a physical simulation of a given artistic medium [Cur+97; Hae+07; LXJ12], procedural techniques [Bou+06; Bou+07; Bén+10; Mon+18], or compositing predefined pen [Sal+97; Pra+01; Sna+06] or brush strokes [Lit97; HE04; Sch+11; ZZ11]. While these approaches give impressive results on the respective domains that they are designed for, they are invariably limited to a single style or a small set of styles, and suffer from unintuitive controls that make it difficult to express artistic intent.

A more modern take on this problem are methods based on generative adversarial networks [Goo+14], which can be trained to perform image-to-image [Iso+17; Zhu+17a; Zhu+17b] as well as video-to-video [Tul+18; Wan+18b] translation, including stylization. Researchers have also introduced neural network based approaches that target artistic stylization specifically [JAFF16; UVL16; Uly+16b; Wan+17; UVL17; WRB17], training one network per style. These methods cannot reproduce styles that they are not trained on, and for the styles they support, the results typically do not accurately reproduce fine textural details. Sanakoyeu et al. [San+18] attempted to improve the stylization quality by introducing a style-aware content loss, but the results still have some semantic inconsistencies (see supplementary material). Researchers have also introduced stylization techniques that transfer arbitrary visual styles to content images using a single network at the expense of limited faithfulness to the target styles [HB17; Li+17]. In general, neural approaches require time-consuming and arcane training process and offer limited user control [Gat+17].

Example-based approaches naturally support stylization using arbitrary style imagery, and no training is needed. The most widespread approach formulated the concept of Image Analogies [Her+01], where guidance channels are added to both the style exemplar and the target photo to guide a patch-based synthesis algorithm [WSI07; Kas+15; Fi16] which decides how different features of the style should be transferred to various regions of the target. The remaining problem is finding appropriate guidance channels, which can be generated algorithmically in certain cases [Bén+13; Jam+15; Fi16] or for particular content (e.g., faces [Fi17]). Creating the guiding channels manually is possible but unintuitive and highly laborious in the case of video.

To circumvent this problem, generic approaches which do not require specific guidance [GEB16; Fri+16] were formulated. More recent neural-based techniques [LW16c; Lia+17; Gu+18] achieve this by using responses of the VGG network trained on object classification [SZ14] to guide the synthesis. These latter approaches produce impressive results when used on images structurally similar to those in ImageNet – natural photographs with a single identifiable foreground object or scene – but are difficult to control and behave unpredictably when generalizing to different types of images such as complex natural scenes or paintings of abstract styles.

Stylization of video offers the additional challenge of handling temporal coherence. This was itself a topic of previous research, where coherence was formulated as an additional constraint for patch-based synthesis together with the control over the amount of visible temporal flickering [Fi14; Fi17; Dvo+18]. Similarly, for generic style transfer not requiring specific types of guidance, explicit temporal coherence was incorporated into neural-based [Che+17; Gup+17; San+18; RDB18] as well as patch-based [Fri+19] techniques. Lai et al. [Lai+18] introduced a blind temporal coherency approach that takes per-frame stylized video as input and outputs a temporally consistent video as post-processing.

We based our approach on the image analogies framework that offers both precise control as well as the ability to handle arbitrary style. We combine keyframe-based user control as in the method of Bénard et al. [Bén+13] with a synthesis process similar to that used in the approach of Fišer et al. [Fi17]. A key added value of our solution is that we overcome two significant drawbacks of these previous methods: (1) dependence on a specific target domain (3D computer-generated animation and facial video) and (2) inability to handle challenging scenario when multiple inconsistent keyframes are used to stylize the target sequence. To do that we design a new set of domain-independent guidance channels and formulate a corresponding error metric for the subsequent patch-based synthesis. To combine content from multiple keyframes, we propose a solution that prefers high-frequency details according to their relevance and avoids loss of contrast.

5.3 Our Approach

The input to our method is a *target* video sequence T and one or more stylized keyframes or *style exemplars*, S. To create keyframes, artists can paint digitally or physically using their preferred artistic media over arbitrarily-selected frames of a video. Similar to the physical painting process used in StyLit [Fi16], we print a low-contrast version of the

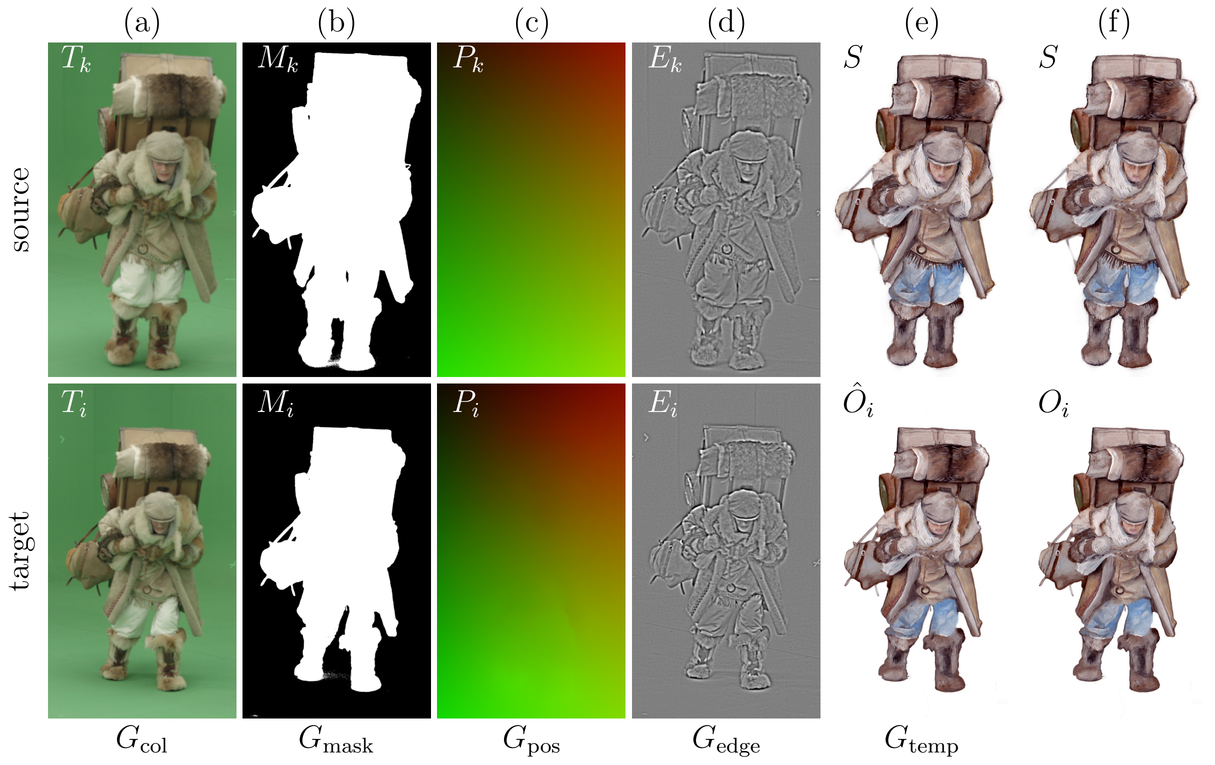


Figure 5.3: *The set of guidance channels used by our method. G_{col} is essential to preserve the target appearance, G_{mask} helps to preserve sharp object boundaries, G_{pos} is important to maintain overall structure, G_{edge} makes synthesis more robust to illumination changes and improves positioning of stylized features, G_{temp} is essential for temporal coherence. For further details, please refer to the text. Video frames (T_k and T_i) courtesy of © MAUR film, style exemplar (S) courtesy of © Pavla Sýkorová, used with permission.*

frame with registration marks, which allows accurate re-digitization and registration of stylized artwork. The output of our method is a temporally coherent video sequence O, in which every frame is stylized analogically to the style exemplar S, i.e., various semantic parts of the input sequence are stylized the same way as in the example frame.

One possible approach to example-based video stylization is to estimate dense correspondences between the target keyframe T_i and all other frames in the sequence [HaC+11; Yüc+12], and then use the resulting deformation field to warp the style exemplar. However, this naive approach would introduce undesirable texture distortion to the example style and generate artifacts in situations like disocclusions or lighting changes in the target sequence. To address this fundamental drawback we formulate our problem as a guided patch-based synthesis similarly to [Bén+13; Fi17]. However, since in our scenario we do not have any prior knowledge of the underlying scene we need to design a new set of guiding channels that can be computed solely based on the input video.

For clarity, we first explain the stylization process with just one keyframe and then show how it extends to multiple keyframes.

5.3.1 Guidance for a single keyframe

Our new set of guiding channels consists of the original video frames G_{col}, mask G_{mask}, positional G_{pos}, edge G_{edge}, and temporal G_{temp} guides (see Fig. 5.3). These will be explained next.

Color guide G_{col} corresponds to the original color frames of the target sequence T (see Fig. 5.3a). It captures appearance changes, e.g., facial gestures, subtle cloth deformations, varying illumination, etc.

Mask guide G_{mask} highlights the objects of interest. It helps the algorithm distinguish object boundaries to handle occlusion and also allow for layered stylization if preferred by artists. When there is no strong occlusion in T or no need to accurately delineate object boundaries, addition of the mask guide is optional otherwise G_{mask} can be obtained using, e.g., green screen matting (see Fig. 5.3b), color separation or other semi-automatic segmentation method [LW16b].

Positional guide G_{pos} helps the algorithm maintain the overall structure of the stylized keyframe for meaningful transfer (see Fig. 5.3c). It serves to resolve ambiguity between distinct features which have similar appearance, but need to be stylized differently as artist desired. In Fig. 5.5a,b the result of synthesis without using G_{pos} is visible. Note, how the light brown wood texture from behind the subject shows up on the leather bag. We define G_{pos} as a dense correspondence map between the current frame T_i and the keyframe T_k. We compute this map by first estimating optical flow between consecutive frames of T using SIFT Flow [LYT11]. This yields a sequence of inter-frame motion fields D_i, which we use to incrementally propagate the original pixel coordinates encoded in a coordinate map P_k (see Fig. 5.4a). We only perform this advection on the pixels inside the object mask M_k (Fig. 5.4b), and use diffusion [Orz+08] to smoothly fill in the remaining values (Fig. 5.4c,d). The resulting map could introduce considerable texture distortion if used directly to warp stylized keyframes (see supplementary material). However, when used as a guide for patch-based synthesis, it encourages transfer of correct style features to the intended locations. Singularities and distortions that would result from direct advection are prevented by the other guiding terms, c.f., error metric (5.1).

Edge guide G_{edge} highlights the object edges and salient features in the target sequence (see Fig. 5.3d), making the result less volatile with respect to color variation in G_{col} caused especially by changes in illumination. Because many artistic styles emphasize edges, this term has the additional benefit of "anchoring" appropriate style features (see Fig. 5.5c,d). We define $G_{\text{edge}}(T_i) = T_i - \mathcal{N}_\sigma \circ T_i$, where $\mathcal{N}_\sigma$ is a Gaussian filter with standard deviation σ.

Temporal guide G_{temp} is designed to encourage temporal coherence by penalizing the synthesis from diverging too much from a previously synthesized frame [Jam+15; Fi17] (see Fig. 5.3e). We compute G_{temp} by advecting the stylization result of the previous frame O_{i-1} using the motion field D_i computed previously for G_{pos}. The advection produces a stylization prediction $\hat{O}_i$ which is not a satisfactory result on its own due to texture distortion, but as a guide, encourages temporally coherent stylization.

Error metric The set of guiding channels we discussed thus far $\mathbb{G} = \{\text{col}, \text{mask}, \text{pos}, \text{edge}, \text{temp}\}$, defines a patch error measure that is plugged into the original StyLit algorithm [Fi16]. We use superscript $\mathcal{S}$ to denote the source part and $\mathcal{T}$ the target part

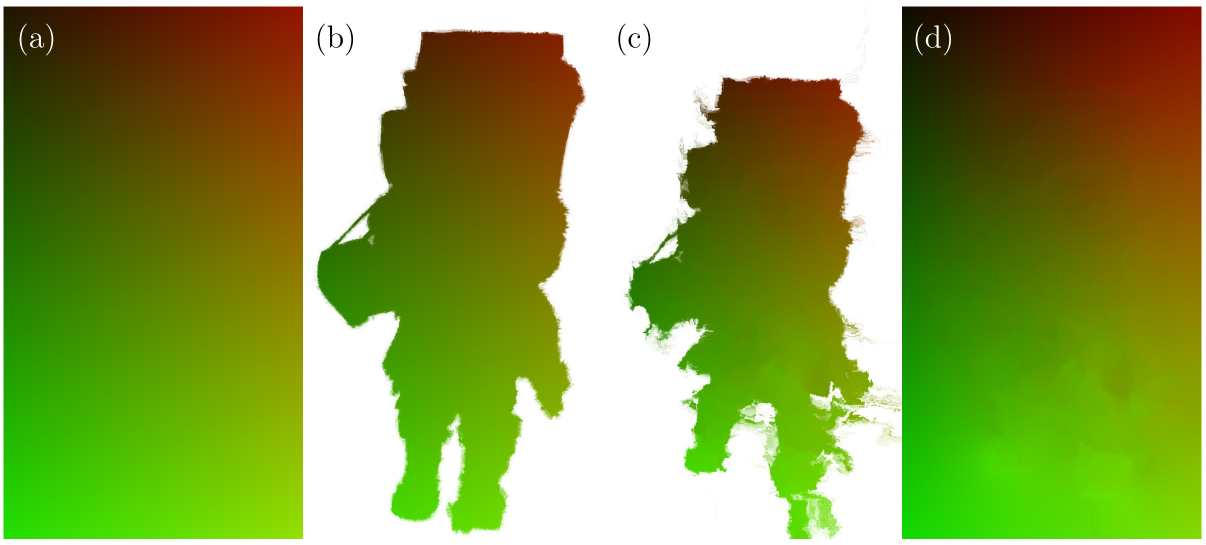


Figure 5.4: *Generating the G_{pos} guiding channel. Red and green color channels denote x and y coordinates. G_{pos} corresponding to the keyframe is constructed as a linear gradient in x - red, and y - green (a). Mask G_{mask} is then applied to the G_{pos} of the keyframe (b). Masked values are then propagated through the sequence according to the motion field D (c). Values outside of the mask are filled using diffusion [Orz+08] (d).*

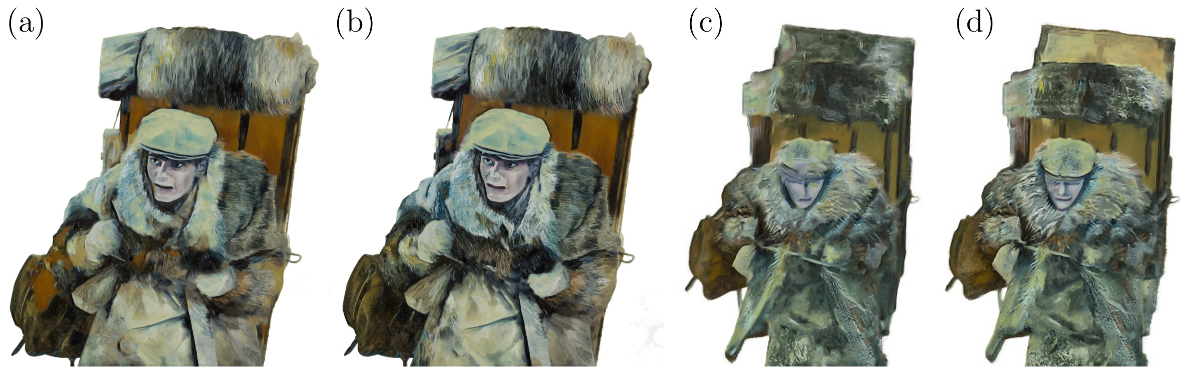


Figure 5.5: *Importance of the G_{pos} and the G_{edge} guidance channels. (a) without the G_{pos}, some features with similar appearance cannot be fully distinguished. Note that the light brown wood texture from box on the subject's back appears on the leather bag at the bottom left corner. The G_{pos} term in (b) helps preserve the overall structure of the different features well. The G_{edge} (c-without, d-with) makes the synthesis less sensitive to illumination changes (see differences in stylization on top of the box) and helps preserving boundaries between the individual style features, thus making the result sharper.*

of each guiding channel. The error metric for matching two patches $p \in \mathcal{S}$ and $q \in \mathcal{T}$ is then computed as follows:

$$E(S, O_i, G^{\mathcal{S}}, G^{\mathcal{T}}, p, q) = ||S(p) - O_i(q)||^2 + \sum_{g \in \mathbb{G}} \lambda_g ||G_g^{\mathcal{S}}(p) - G_g^{\mathcal{T}}(q)||^2 \qquad (5.1)$$

where λ_g is a weighting factor for each individual guiding channel and the first term helps to preserve *texture coherence* by directly matching colors in patches of stylized keyframe S to those in the output frame O_i (see Fig. 5.3f). The style S and all guiding channels remain unchanged during the synthesis. Only O_i is iteratively updated. See [Fi16] for more details about the optimization.

5.3.2 Handling multiple keyframes

In many cases, it is sufficient to have only one keyframe. If, however, a sequence has new content appearing which did not exist in the keyframe and was not stylized, the artist may choose to specify a new keyframe to precisely control the stylization of the new content. The use of multiple keyframes introduces difficulty to the algorithm, since manually created keyframes will inevitably have subtle inconsistency in structure and colors. Previous approaches [She+10; Dar+12; Bro+14] either suffer from detail clutter or produce temporal artifacts such as unnatural "boiling" or "pumping".

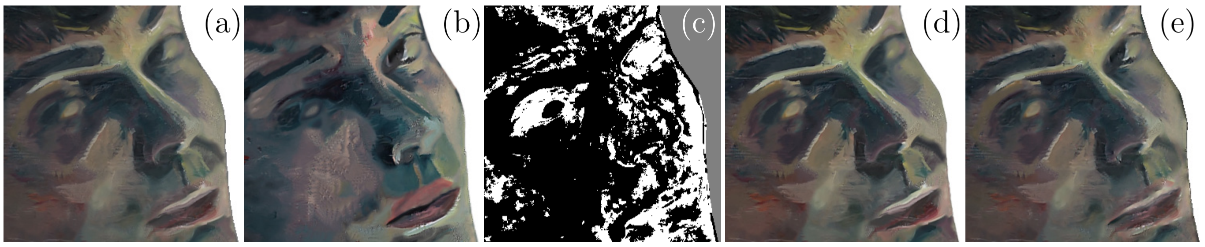


Figure 5.6: *Gradient domain mixing. Two stylized images O_i^a (a) and O_i^b (b) are synthesized at frame i using two different keyframes S_k and S_l. We compute a pixel selection mask Z_i (c) where black pixels indicate the locations where synthesis error E_i^a is lower than E_i^b and white pixels vice versa (gray indicates background). We then pick gradients according to Z_i (∇O_i^a for the black pixels and ∇O_i^b for the white ones) and run a screened Poisson solver on the contrast-preserving blend O_i^{ab} of O_i^a and O_i^b (d). Note that the resulting image O_i (e) contains high-frequency details according to Z_i.*

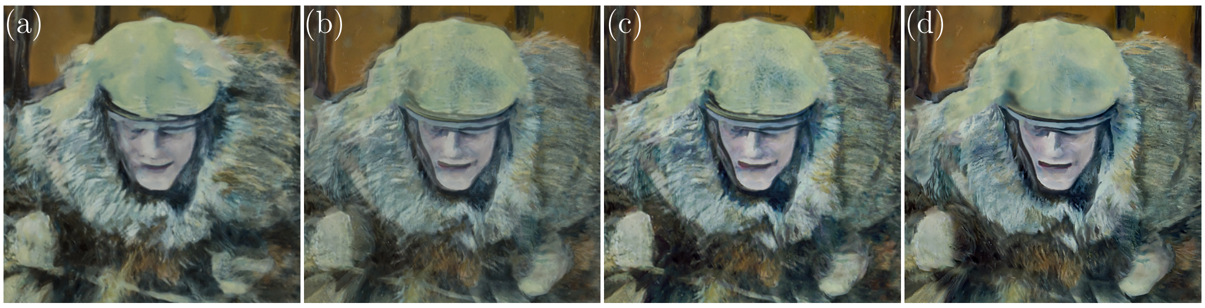


Figure 5.7: *Comparison of different blending methods. (a) Regenerative Morphing [She+10] exhibits some detail clutter and loss of detail. (b) Linear blend leads to contrast loss and ghosting is visible. (c) The contrast-preserving linear blend [HN18] has higher contrast, but the ghosting is still apparent. (d) Our approach has high contrast and ghosting is significantly suppressed.*

We propose a different solution which keeps the keyframe stylization unchanged while producing smooth and seamless transitions between keyframes. We first stylize the sequence using keyframes at the beginning (S_k^a) and at the end (S_l^b) to produce two separately stylized sequences O^a and O^b. To produce the final frame of index i, we blend the corresponding frames O_i^a and O_i^b. Now the question becomes what blending technique should we use.

A trivial approach would be to perform a linear blend: $O_i = (1-\alpha)O_k^a + \alpha O_l^b$, where $\alpha = (i-k)/(l-k)$. Such a solution flattens the original contrast and introduces ghosting artifacts (see Fig. 5.7b). In addition, linear blending implicitly assumes that the content of the frame changes smoothly in time; such an assumption is violated when there is disocclusion in the sequence, which suddenly introduces new local content that exists in keyframe S_l^b but not in S_k^a. In this case, we should stylize the new content using S_l^b exclusively, without blending in any features from S_k^a. To achieve this, we take advantage of the fact that our algorithm gives a patch matching error (5.1) for each pixel p in every frame for both O^a and O^b. Our intuition is that between two patches from O^a and O^b located at pixel p, the one with lower matching error will lead to "better" result and thus should be locally preferred.

Error-based gradient domain fusion In order to merge the best content from the two stylized sequences, we use gradient domain fusion similar to that used in Image Melding [Dar+12] (see Fig. 5.6), where a screened Poisson equation [Bha+08] is applied to perform this task. Our solution, differs in how we select the gradient and how the screening value for reconstruction is computed. For the gradient, we select $\nabla O_i^a(p)$ or $\nabla O_i^b(p)$ according to a pixel selection mask Z_i (see Fig. 5.6) where white pixels indicate the state where the synthesis error $E_i^a(p)$ is lower than $E_i^b(p)$ and thus $\nabla O_i^a(p)$ is selected, while the opposite holds for black pixels. This ensures the blending result borrows the structure and high-frequency content from the synthesis result that most closely matches its respective keyframe.

Preserving color histogram To ensure the global color histogram varies smoothly over time, we use a blended sequence O^{ab} for screening. Instead of linear blending, we use contrast-preserving blending from Heitz and Neyret [HN18], which blends two images O_i^a and O_i^b and produces an image O_i^{ab} with a prescribed histogram H which is constructed by tabulating the colors of pixels according to pixel selection mask Z_i, i.e., we count the colors from pixels which have lower synthesis error. Though in O_i^{ab} ghosting artifacts still exist (see Fig. 5.7c), they are suppressed by the screened Poisson reconstruction in the resulting frame O_i. The screening value O_i^{ab} only serves to regularize the color histogram of the result.

Temporal coherence of pixel selection mask Although the synthesis error usually increases as the target frame gets further away from the keyframe, the increase in error might not be monotonous in some local regions. This behavior may introduce visible flickering since the matching error constraint may cause the algorithm to frequently alternate between choosing contents stylized from different keyframes S_a and S_b. To avoid such temporal instability, we explicitly enforce temporal coherence of the pixel selection mask Z (see our supplementary material for illustrative figure). We store pixel selection mask Z_{i-1} from the previous frame and use estimated inter-frame motion field D_i to produce an initial $\hat{Z}_i$ which indicates existing pixel selection advected from the previous frame. We then update $\hat{Z}_i$ using the lower error constraint as described previously. However, we prevent the situation where a pixel that has already been assigned to take color and gradient information from an image stylized using the later keyframe S_b from switching back to the earlier keyframe S_a.

After applying this refinement, the resulting fused image has better contrast and less ghosting artifacts (Fig. 5.7d).

5.4 Results

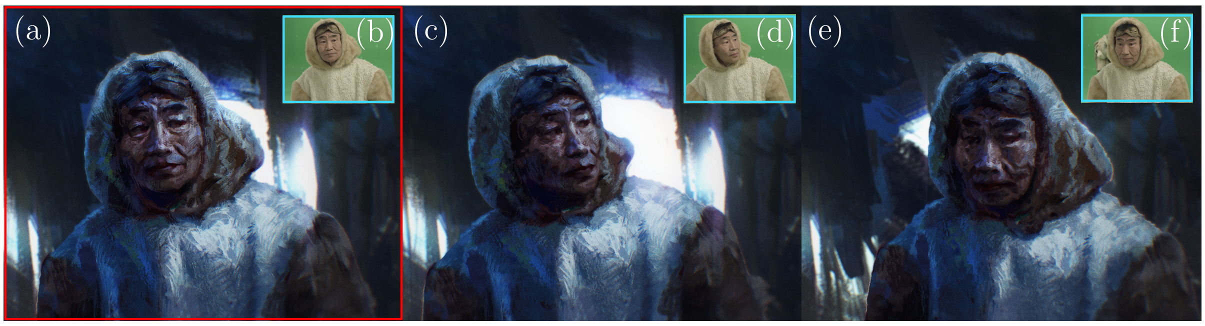


Figure 5.8: *Eskimo sequence: digitally painted keyframe (a) was used to stylize the 148 frames long sequence (b, d, f), stylized frames (c) and (e). Video frames (b, d, f) courtesy of © MAUR film, stylized keyframe (a) courtesy of © Jakub Javora, used with permission.*

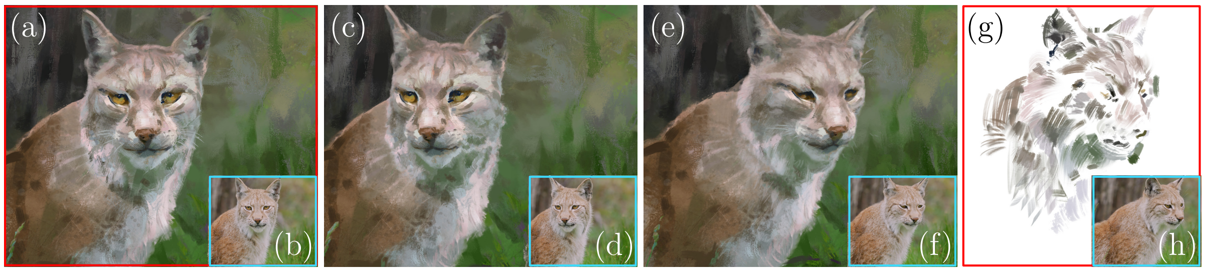


Figure 5.9: *Lynx sequence: digitally painted keyframes (a) and (g) were used to stylize the 100 frames long sequence (b, d, f, h). Keyframe (a) was painted entirely while in keyframe (d) only few strokes were added on top of the synthesis result, stylized frames (c) and (e). Video frames (b, d, f, h) courtesy of © kjekol / Adobe Stock, stylized keyframes (a, g) courtesy of © Jakub Javora, used with permission.*

We pre-process guiding channels off-line on the CPU. This includes green screen matting, optical flow estimation, advection of content from the previous frame, and hi-pass filtering of the target video frame (using $\sigma = 6$). For a one-megapixel frame this process takes less than 20 seconds with the most time-consuming part being the computation of optical flow using SIFT flow method [LYT11]. We use the following default setting of weights for individual guiding channels: $\lambda_{\text{col}} = 6$, $\lambda_{\text{pos}} = 2$, $\lambda_{\text{edge}} = 0.5$, $\lambda_{\text{mask}} = 1$, $\lambda_{\text{temp}} = 0.5$.

The actual synthesis then runs on the GPU (with CUDA) using the StyLit algorithm [Fi16] with the following settings: 5×5 patches, 6 pyramid levels, 12 search-vote iterations, and 6 PatchMatch sweeps [Bar+09]. We also use the optimization described in [Fi17], i.e., the nearest neighbor field propagation is executed only on patches that lower the matching error in previous search step. With this fine-tuning, we can synthesize one-megapixel frame in 9 seconds using GeForce GTX 1070.

For the fusion of sequences stylized from different keyframes we implemented the method of Heitz and Neyret [HN18] as well as screened Poisson solver [Bha+08] where we set the screening parameter $\lambda_d = 0.1$. The computation runs on the CPU and time for merging two one-megapixel frames is on average 10 seconds.

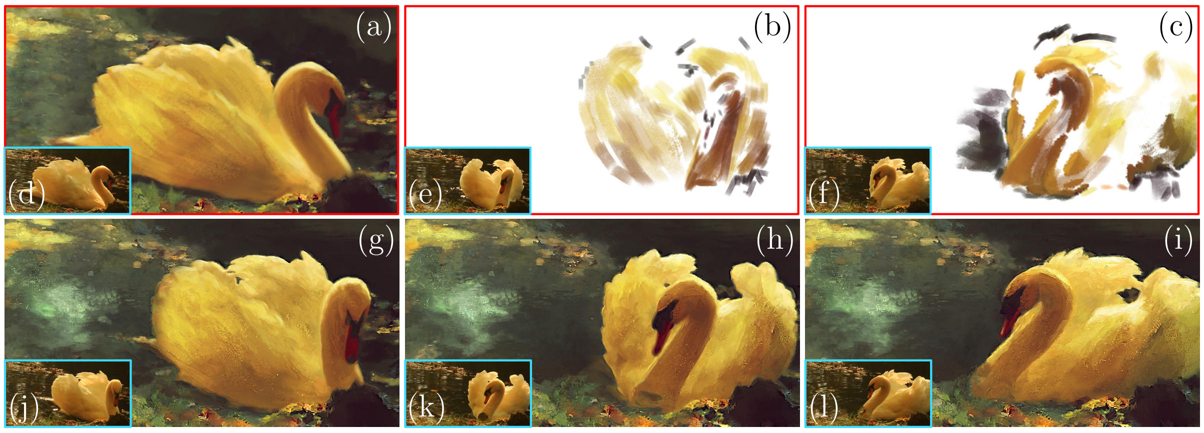


Figure 5.10: *Swan sequence: five digitally painted keyframes out of which three are shown in this figure (a, b, c) were used to stylize the 437 frames long sequence (d, e, f, j, k, l). Keyframe (a) was painted entirely while in keyframes (b) and (c) only few strokes were added on top of the synthesis result, stylized frames (g, h, i). Video frames (d, e, f, j, k, l) courtesy of © Primus1 / Adobe Stock, used with permission.*

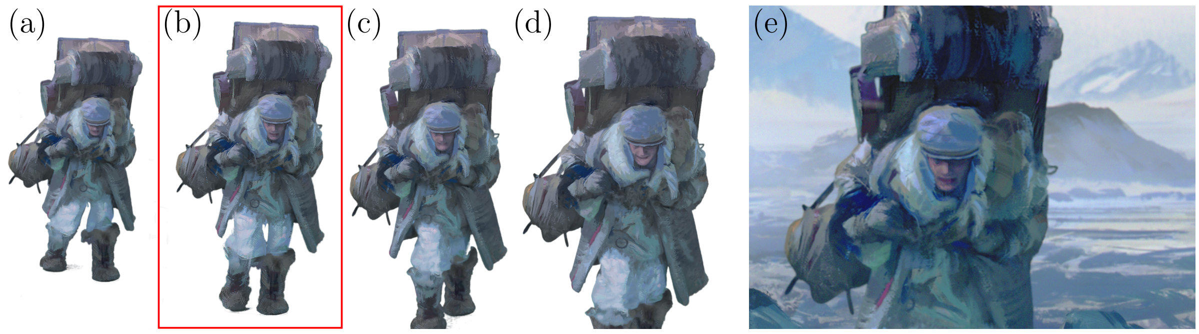


Figure 5.11: *Snowstorm composition: only one keyframe (b) was used to stylize video sequence with 521 frames including frames (a, c, d). The final composition (e). Stylized keyframe (b) and the final composition (e) courtesy of © Jakub Javora, used with permission.*

We validate our approach on multiple sequences from real production (please refer to our supplementary video) with varying complexity using different styles including physical, artistic media such as oil paint, watercolor, pencil drawing, and digital paint. The number of keyframes used for synthesis depends on the shot complexity. One keyframe is typically sufficient for shots where objects move mostly in the camera plane without occlusion or significant changes in illumination (see Figures 5.1, 5.8, 5.11, and 5.12). For more complex shots with out-of-plane rotation and illumination changes, two (Figures 5.9 and 5.14) or more keyframes (Figures 5.10 and 5.13) are necessary. The keyframes painted by an artist are highlighted with red rectangles in the figures. In a fully digital pipeline, not all keyframes need to be prepared from scratch. Instead, one can stylize the entire shot using one painted keyframe, and then manually fix deteriorated regions when needed. Frames with corrections become new keyframes (see Figures 5.9 and 5.10).

For shots with frequent occlusions, we separate each frame into multiple layers for best synthesis quality and lowest number of keyframes (see our supplementary material).

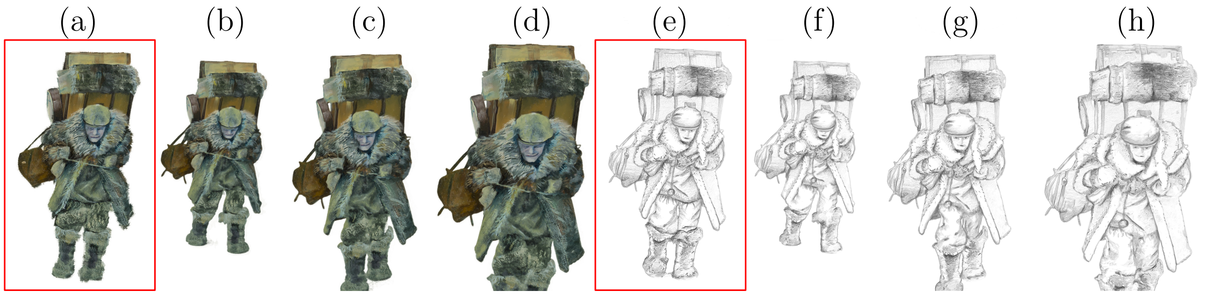


Figure 5.12: *Two different style exemplars—oil paint (a) and pencil drawing (e) were used to stylize the same set of target video frames as in Fig. 5.1. Style exemplars courtesy of © MAUR film, Václav Švankmajer (a), © Pavla Sýkorová (e), used with permission.*

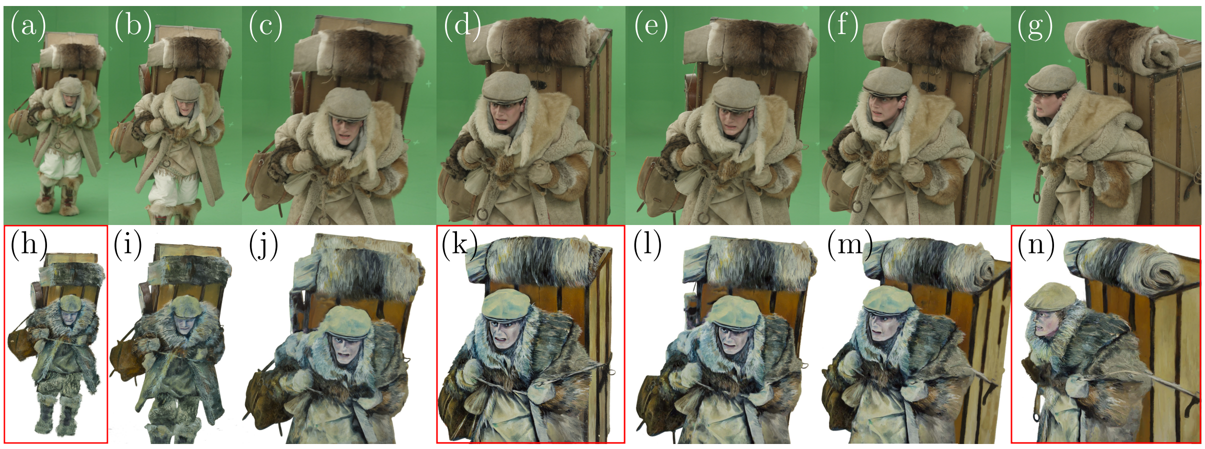


Figure 5.13: *Long video sequence with multiple keyframes: (a–g) are the target frames, (i, j, l, m) are resulting synthesized frames using two respective nearest keyframes (h, k, n). In total, the sequence contains 889 frames and 8 keyframes. Video frames (a–g) and stylized keyframes (h, k, n) courtesy of © MAUR film, Václav Švankmajer, used with permission.*

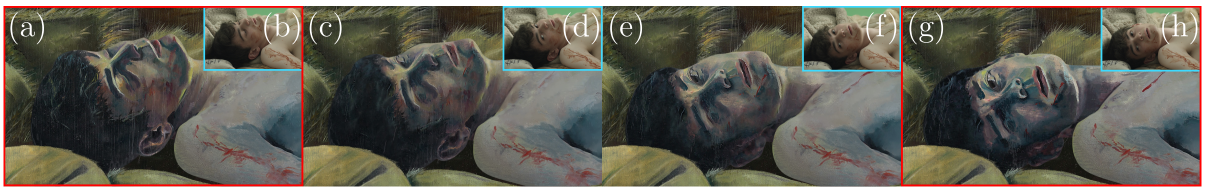


Figure 5.14: *Stylization between two keyframes: target video sequence (b, d, f, h) is first stylized using keyframe (a), then the same sequence is stylized using keyframe (g), and finally, the two resulting stylized sequences are then fused together (c, e). To stylize 1545 frames, only two keyframes were used. Video frames (b, d, f, h) and stylized keyframes (a, g) courtesy of © MAUR film, Václav Švankmajer, used with permission.*

We compared our approach with the stylization framework proposed by Bénard et al. [Bén+13] (see Fig. 5.15b). Although the original method does not support generic video stylization, we prepared the necessary guiding channels using our technique and

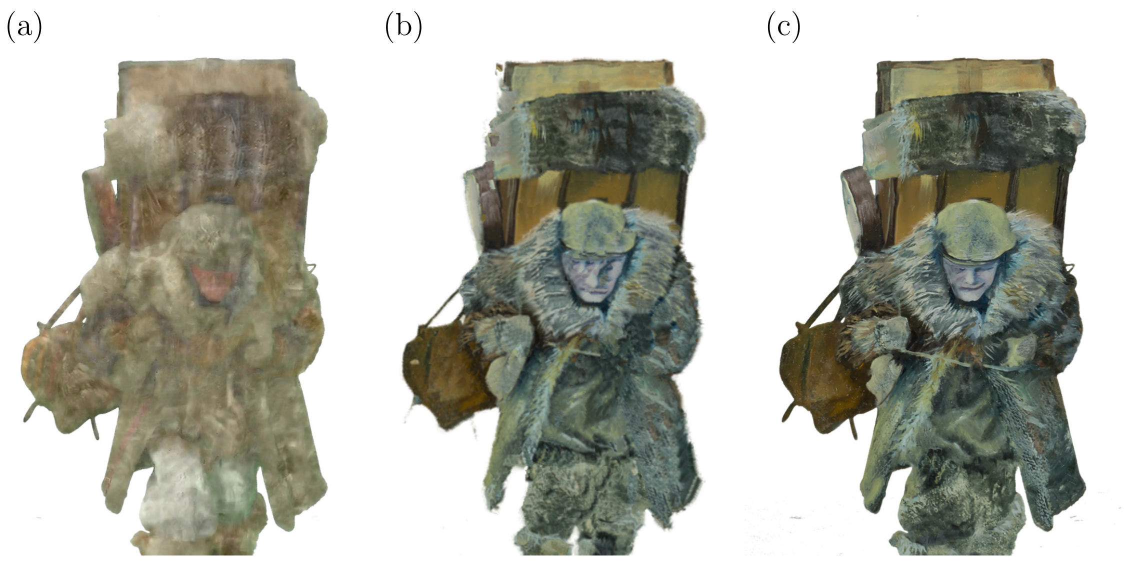


Figure 5.15: *Comparison with patch-based techniques: (a) Frigo et al. [Fri+19], (b) Benard et al. [Bén+13], (c) our approach.*

provide them as an input to their algorithm. We also compared with another patch-based technique that supports temporal coherence [Fri+19] (see Fig. 5.15a). See our supplementary video for comparison on the entire sequence. We were interested in how well each algorithm preserves the quality of the original style exemplar and handles temporal coherence. From the results, Bénard et al.'s method has difficulty in preserving high-frequency details of the original style exemplar and tends to produce visible drifting chunks resulting in more temporal noise. Frigo et al.'s method also fails to preserve sharp details and cannot reproduce the colors in the style exemplar.

We also performed a comparison with state-of-the-art neural-based approaches (see Fig. 5.16 and our supplementary video). The method of Ruder et al. [RDB18] preserves temporal coherence, but does not fully transfer the details of the style exemplar. The method of Li et al. [Li+17] has similar appearance problems, and does not support temporal coherence. The approach of Liao et al. [Lia+17] better reproduces the style, but introduces visible misalignment of salient features and again, does not preserve temporal coherence. The approach of Gu et al. [Gu+18] can avoid the misalignment at the cost of smoothing out important high-frequency details of the original style exemplar. In addition, the last three methods suffer from severe temporal flickering when applied on video. We tried to post-process all three with the blind temporal consistency method of Lai et al. [Lai+18]. Although the results were a bit temporally smoother, they exhibited additional loss of contrast and detail. See supplementary video.

5.5 Limitations and Future Work

Although our new technique improves visual quality over the state-of-the-art and enables considerable reduction of manual labor in the creation of stylized videos, there are still some limitations that can motivate further research.

One of the key drawbacks of our approach is sensitivity to more substantial illumination changes in the target video. This may happen, e.g., when a part of the stylized object

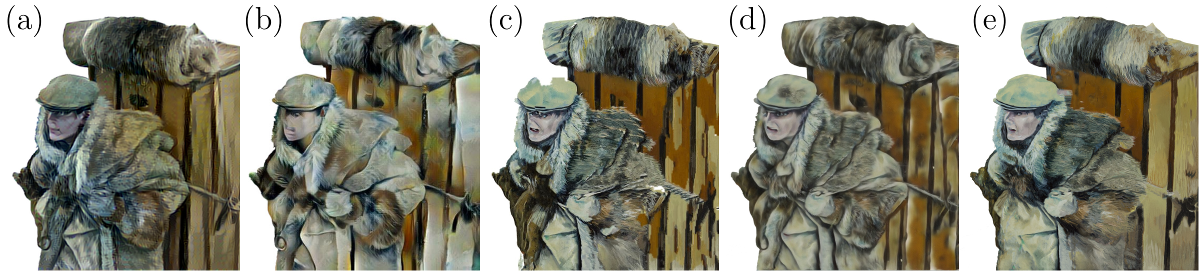


Figure 5.16: *Comparison with recent neural-based methods: (a) Ruder et al. [RDB18], (b) Li et al. [Li+17], (c) Liao et al. [Lia+17], (d) Gu et al. [Gu+18], (e) ours. Fig. 5.13k was used as the style exemplar for all methods.*

is originally in light, and then it enters a shadow. In this case, the inconsistent colors of G_{col} can be misleading. Although the use of G_{edge} and the diffuse studio lighting may suppress this behavior (see Fig. 5.5c,d and our supplementary material), a more advanced appearance matching technique would be helpful. Fišer et al. [Fi17] used the method of [Shi+14] that is, however, tailored to facial images. A more generic approach is needed in our scenario.

As a related problem, structural changes between nearby keyframes harm the synthesis quality. In some cases, separation into layers may help to reduce clutter and preserve content coherence. However, the appearance of target objects may change considerably if they contain dynamic high-frequency structures (e.g., distinct texture or wrinkles on clothing, see our supplementary material). This change will lead to inconsistencies in G_{col}. In these scenarios, more appropriate clothing or an additional detail-removing filtering [Xu+11; BHY15] may help improve the synthesis quality.

Although our technique for mixing two stylized sequences does well in preserving contrast and suppressing ghosting, excessively large structural changes may still lead to subtle ghosting effect due to usage of blended screening target (see Fig. 5.7c, eyebrow in Fig. 5.14, and our supplementary material). Though solutions exist that can deform local features for better structural matching [RSK10; Lia+14], we cannot apply them since we need to avoid free-form deformations that may destroy the structure of the original paint texture. A better warping scheme that preserves local high-frequency structure could potentially improve our method's tolerance to these large structural changes.

5.6 Conclusion

We presented a new approach to temporally coherent artistic stylization of video. Our two primary design considerations were (1) to allow direct and free-form artistic control in the form of keyframes painted in any desired traditional medium and (2) to support stylization of arbitrary input videos. Our approach enables a practical pipeline in real production shots for creating traditional-style animation from live-action performance capture. It further provides an easier artistic video creation workflow eliminating the need for a tedious frame-by-frame painting process while preserving the unique and rich visual qualities of traditional artistic media. We hope this will help bridge the gap between live action, 3D animation, and traditional hand-painted animation.

Chapter 6

Stylization Using Few-Shot Patch-Based Training

6.1 Introduction

Example-based stylization of videos became recently popular thanks to significant advances made in neural techniques [RDB18; San+18; Kot+19a]. Those extend the seminal approach of Gatys et al. [GEB16] 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 [Bén+13; Jam+19]. Those techniques employ guided patch-based synthesis [Her+01; Fi16] 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., [She+10; Jam+19]), 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,

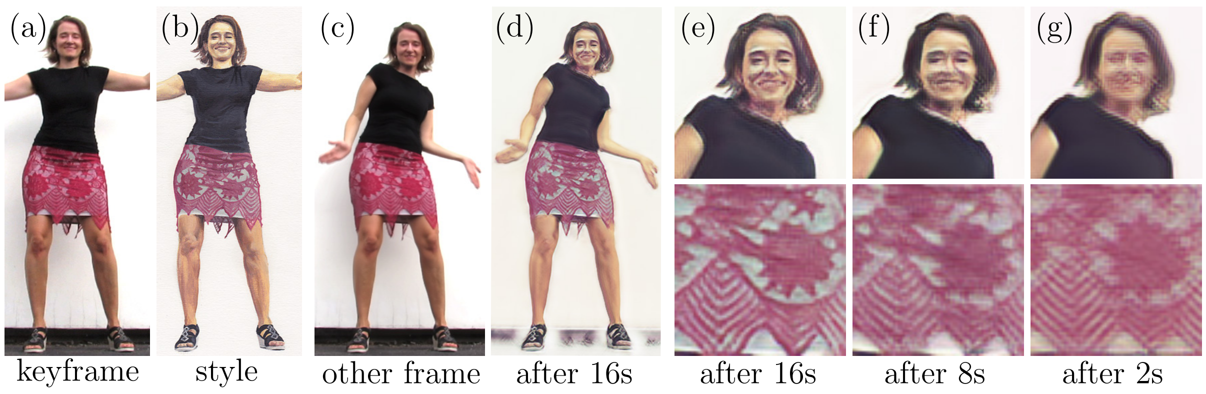


Figure 6.1: *An example of a sequence stylized using our approach. One frame from the original sequence is selected as a keyframe (a) and an artist stylizes it with acrylic paint (b). We use this single style exemplar as the only data to train a network. After 16 seconds of training, the network can 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.*

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 [Iso+17; Wan+18a], 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+19; Wan+19b] 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.

6.2 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 [WJE19; Li+19] or compute an optical flow [Che+13] between consecutive frames, and use it to propagate the stylized content from the keyframe. However, as shown in Jamriška et al. [Jam+19] 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. [Bén+13] who use guided patch-based synthesis [Her+01] 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. [Jam+19]. 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 [Iso+17] could provide a more appropriate solution. Once trained, they can perform semantically meaningful appearance transfer in real-time as recently demonstrated on human portraits [Fut+19]. 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 [Wan+18a; Cha+19] which is completely intractable.

Recently, few-shot learning techniques were introduced [Wan+19a; Wan+19b] 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 [LW16c; Uly+16a], 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 [SCI18] or to maintain high-level context [Zho+18; Sho+19; SDM19]; 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 [Fi14] 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 [Bén+13; Fi17; Jam+19; Fri+19] or neural-based [Che+17; San+18; RDB18], 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+18] 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. [Mul+19]. 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.

6.3 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. 6.2 for details).

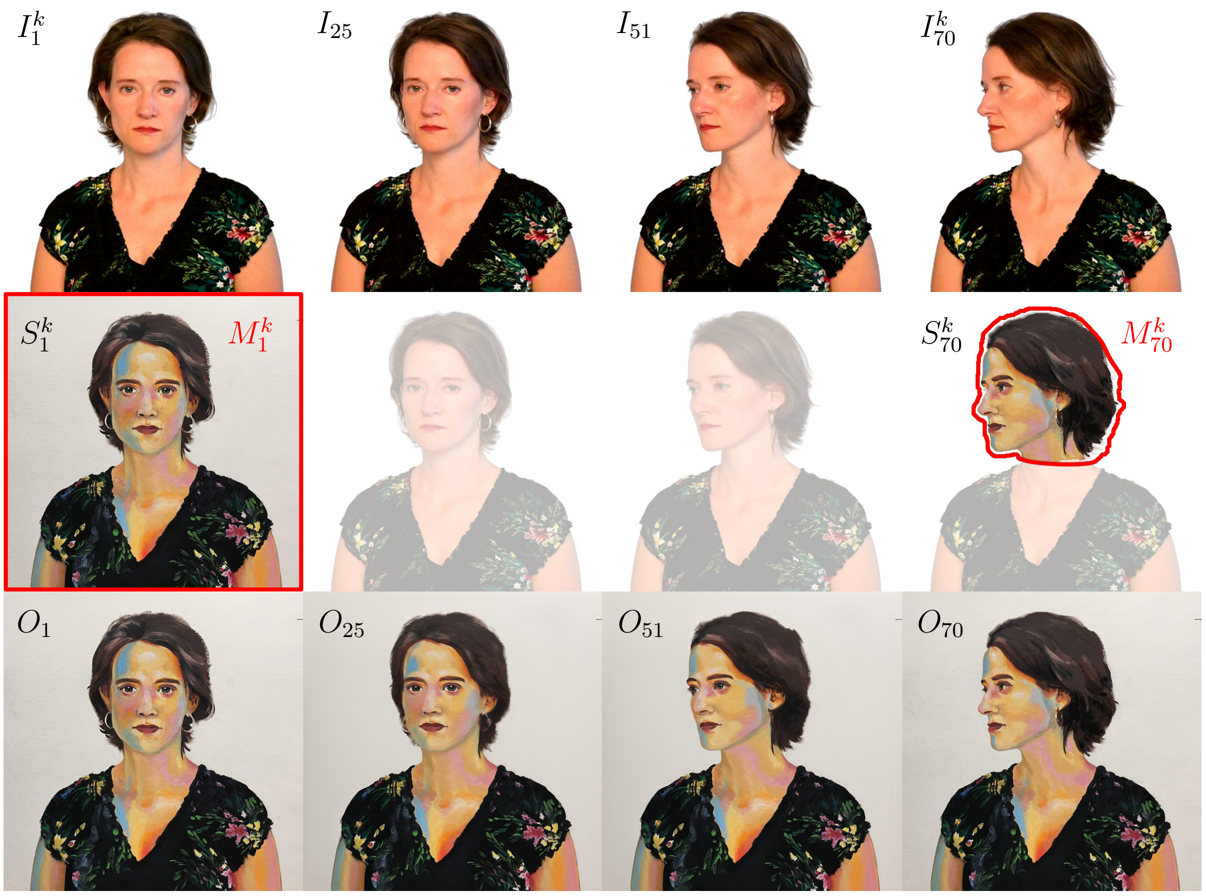


Figure 6.2: *The setting of video stylization with keyframes. The first row shows an input video sequence I. There are two keyframes painted by the user, one keyframe is painted fully (S_1^k) and the other is painted only partially (S_{70}^k). Mask M_1^k denotes that the entire keyframe is used; mask 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 [Jam+19]. 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.

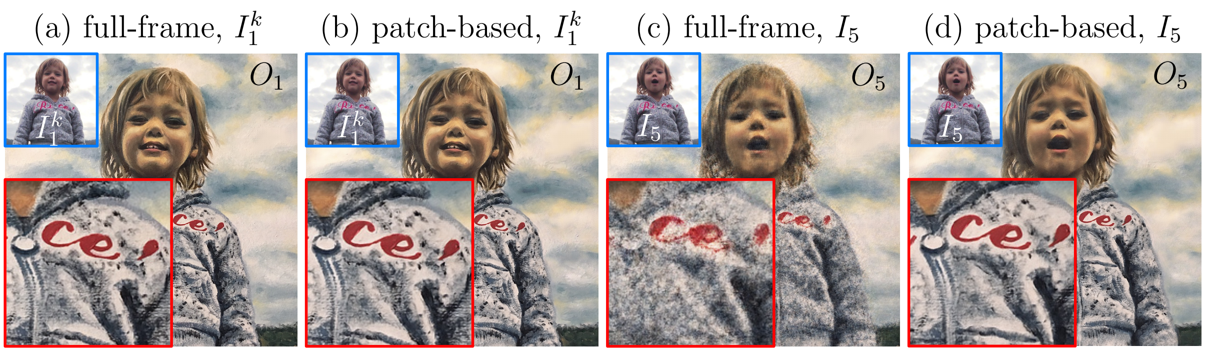


Figure 6.3: *Comparison of full-frame training vs. our patch-based approach: the original frames from the input sequence I are marked in blue and details of their stylized counterparts O are marked in red. The full-frame training scheme of Futschik et al. [Fut+19] (a) as well as our patch-based approach (b) closely reproduce the frame on which the training was performed (see the frame S_1^k in Fig. 6.6). 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.*

To design such a filter, we adopt the U-net-based image-to-image translation framework of Futschik et al. [Fut+19], 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. 6.3c 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 [Fi17]. 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 6.3 and 6.9. 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.

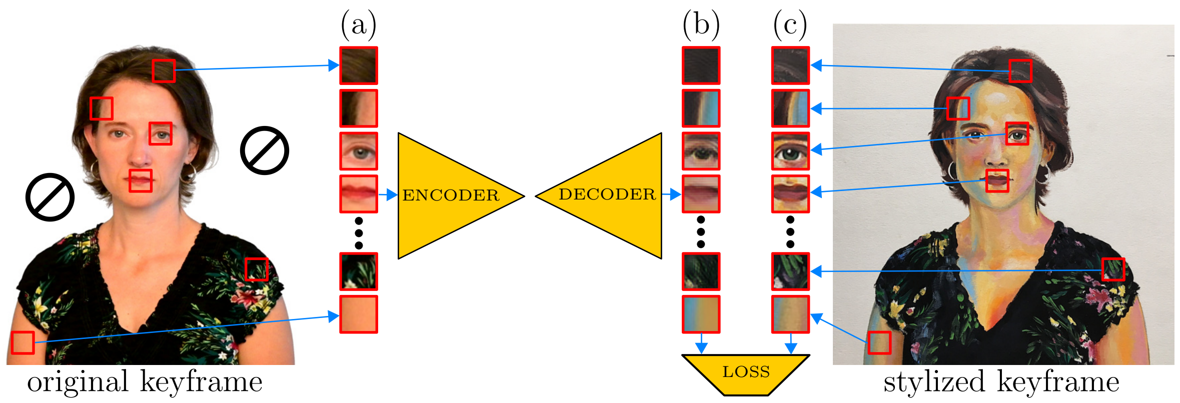


Figure 6.4: *Training strategy: we randomly sample a set of small patches from the masked area of the original keyframe (a). These patches are then propagated through the network in a single batch to produce their stylized counterparts (b). We then compute the loss of these stylized counterparts (b) 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 [Fut+19]. Video frame (left) and style exemplar (right) courtesy of © Zuzana Studená.*

6.3.1 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 [Fut+19], we randomly sample smaller rectangular patches from all stylized keyframes S^k (see Fig. 6.4) 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. 6.5). 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 [Zho+18].

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.

6.3.2 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. 6.8). Besides that, we also need to balance the model size to capture the essential characteristics of

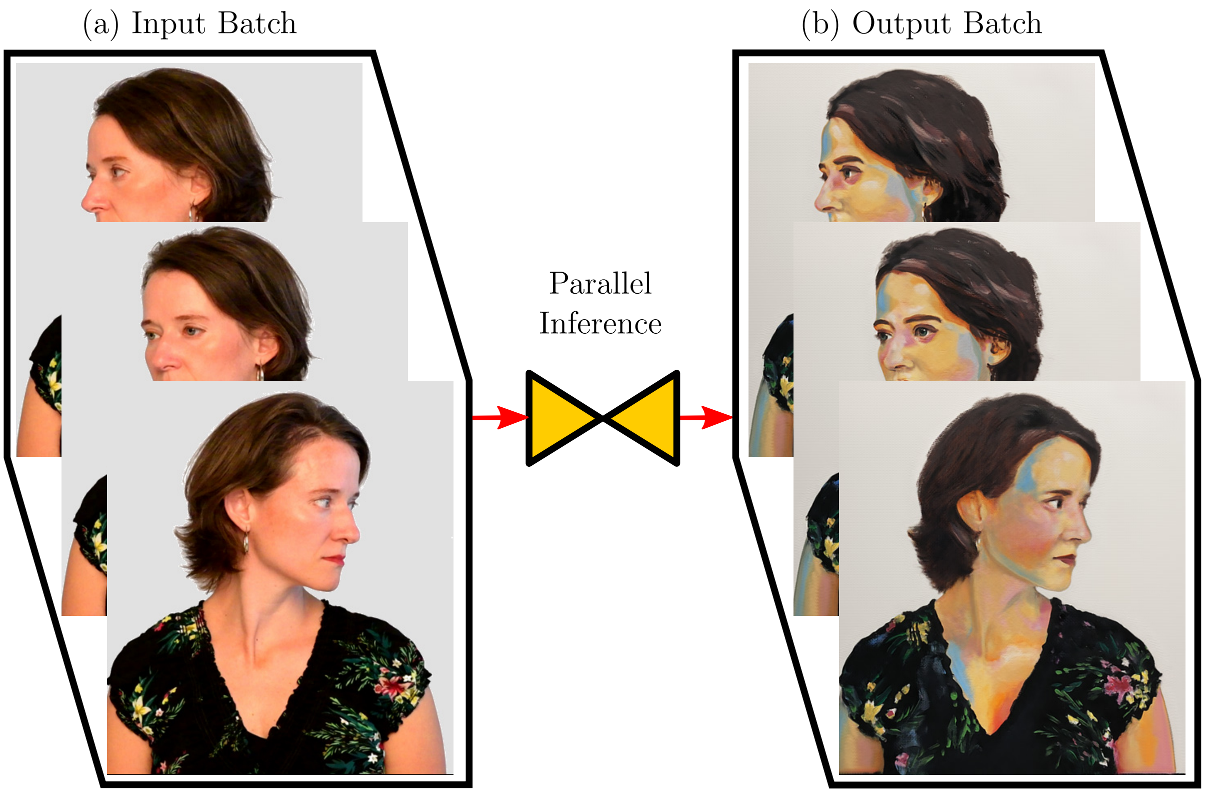


Figure 6.5: *Inference: thanks to the fully convolutional nature of the network, we can perform the inference on entire video frames, even though the training is done on small patches only. Since the inference does not depend on other stylized frames, all video frames can be stylized in parallel or in 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.*

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. [Fut+19] computed over the frames inferred by our network and their counterparts stylized using the method of Jamriška et al. [Jam+19]. 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. 6.6). In-depth elaboration can be found in Section 8.4.

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.

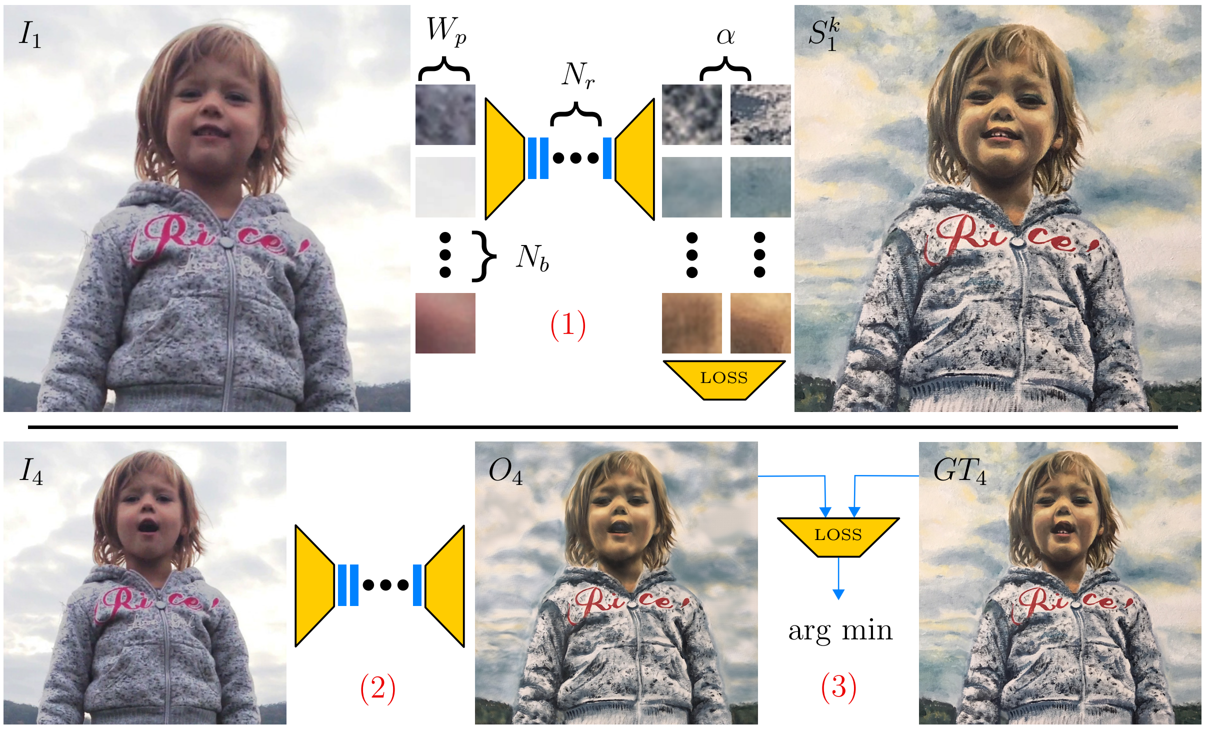


Figure 6.6: *To fine-tune critical hyper-parameters of our network, we propose the following optimization scheme. We tune batch size N_b, patch size W_p, number of ResNet blocks N_r, and learning rate α. Using the grid search method we sample 4-dimensional space given by these hyper-parameters and for 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 [Jam+19] (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.*

6.3.3 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 [BM05]. 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. 6.7). 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 [SDC09]. 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.

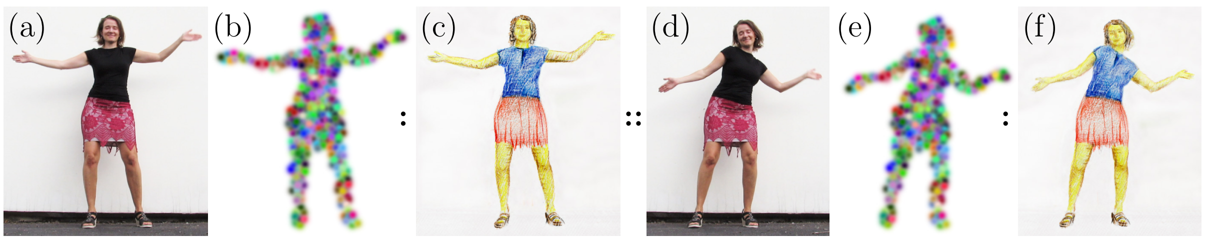


Figure 6.7: *To suppress visual ambiguity of the dark mostly homogeneous T-shirt in (a) an auxiliary input layer is provided that contains a mixture of randomly distributed and colored Gaussians (b). The translation network is trained on patches of which input pixels contain those additional color components. 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.*

The reason why the mixture of Gaussians is used instead of directly encoding pixel coordinates as done, e.g., in [Liu+18; Jam+19] 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).

6.4 Results

We implemented our approach in C++ and Python with PyTorch, adopting the structure of the appearance translation network of Futschik et al. [Fut+19] 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. [Jam+19].

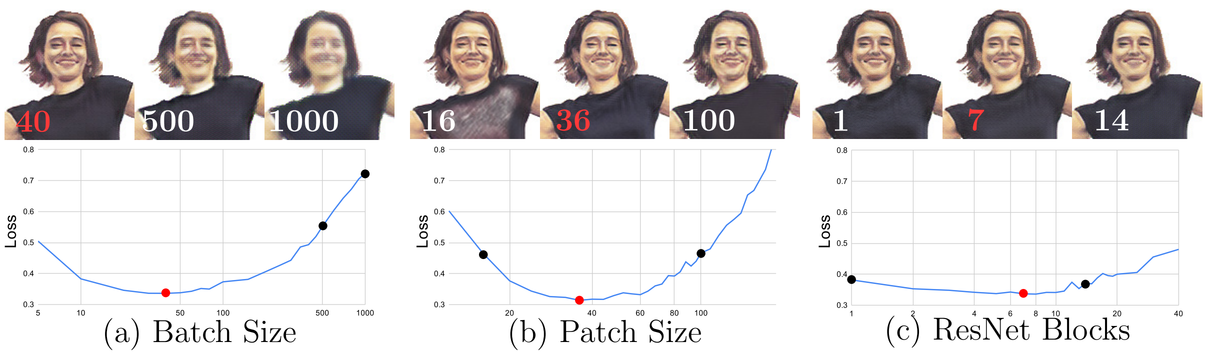


Figure 6.8: *Influence of important hyper-parameters on visual quality of results. The loss, y-axes, is computed w.r.t. the output of Jamriška et al. [Jam+19]. The best setting for each hyper-parameter is highlighted in red: (a) The loss curve for the batch size N_b—the number of patches in one training 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.*

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. [Jam+19] 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. 6.8 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. 6.8a). 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. 6.8b) 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. 6.8c) 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. 6.1).

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 artistic styles (oil paint, acrylic paint, chalk, color pencil, markers, and digital image). Simpler sequences were stylized using only one keyframe (see Figures 6.1, 6.3, 6.7, 6.11, and 6.12) while the more complex ones have multiple (ranging from two to seven, see Figures 6.14, 6.13, 6.15, and 6.16). 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.

6.4.1 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. 6.9).

We compared our approach with the current state-of-the-art in keyframe-based video stylization [Jam+19]. For the results see Figures 6.10, 6.12, 6.14, 6.15, 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 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 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.

6.4.2 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.

6.5 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 [Jam+19] (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. 6.10. 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 [Wan+18b] 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.

6.6 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.

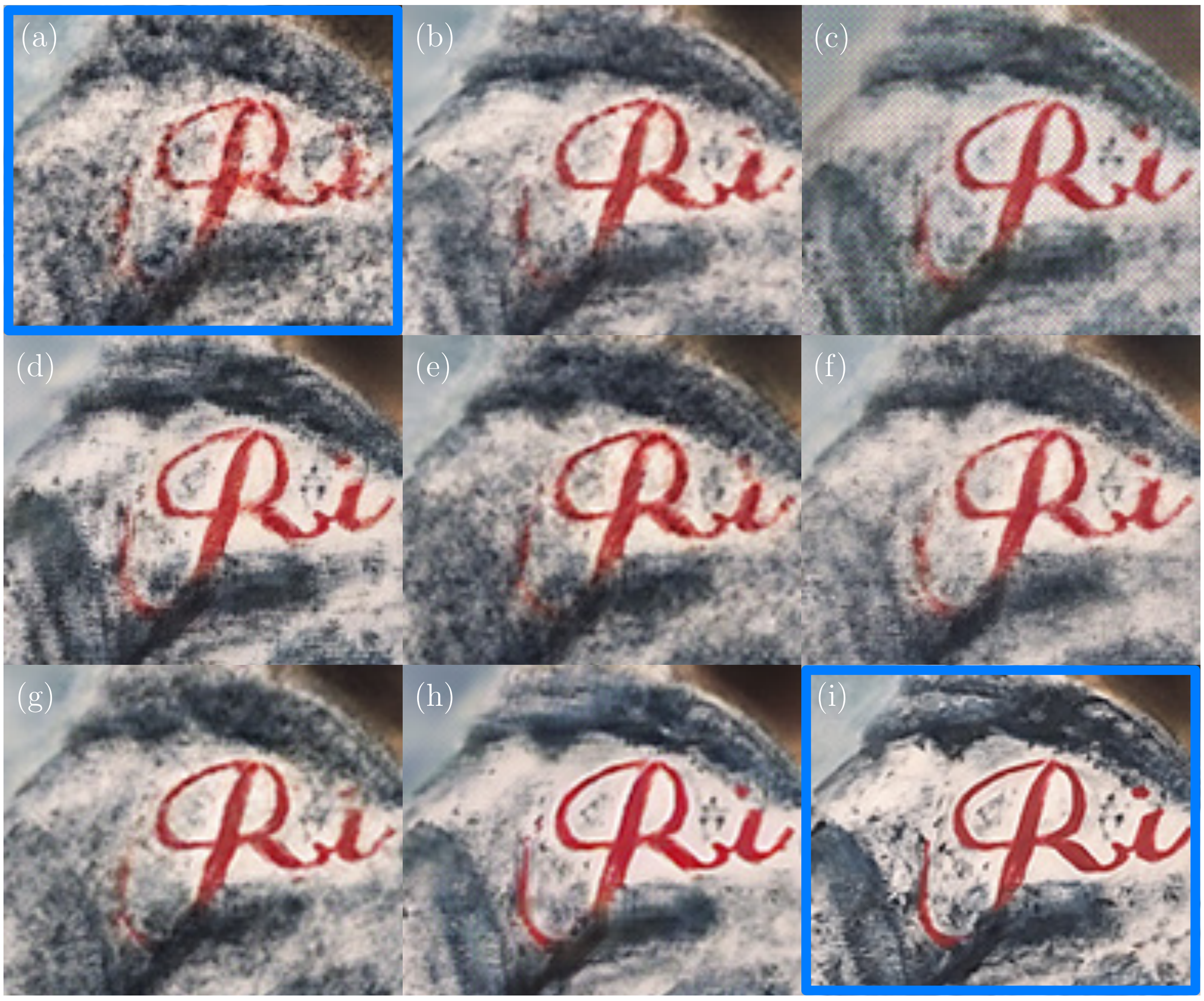


Figure 6.9: *To deal with the overfitting caused by a minimal amount of training data, we tried several commonly used techniques to enforce regularization. In all cases shown in this figure, we trained the network on the first frame; the shown results are zoomed details of the fifth frame. (a) is a result 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.*

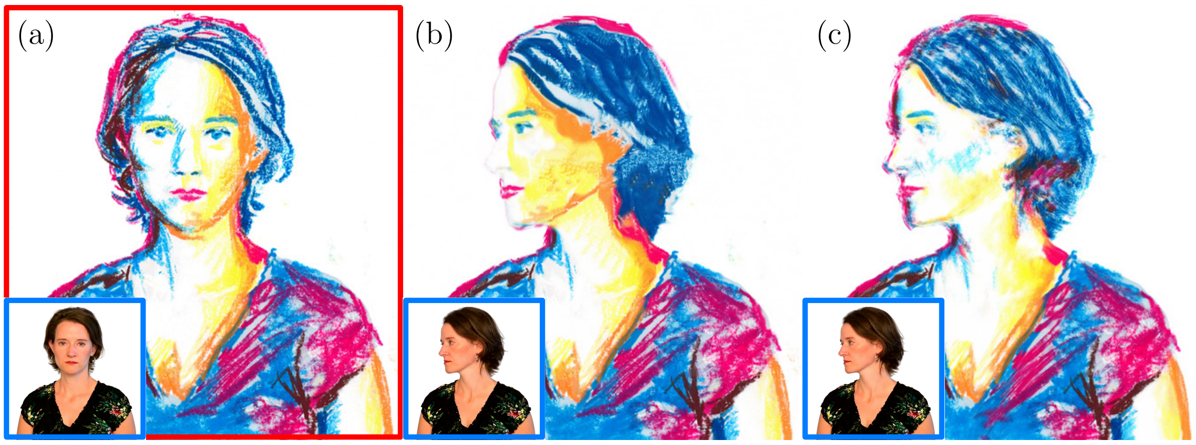


Figure 6.10: *When the target subject undergoes a substantial appearance change, the results of both Jamriška et al. [Jam+19] (b) and our method (c) exhibit noticeable artifacts. The parts that were not present in the keyframe are reconstructed poorly—see the face and hair regions where [Jam+19] produces 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.*

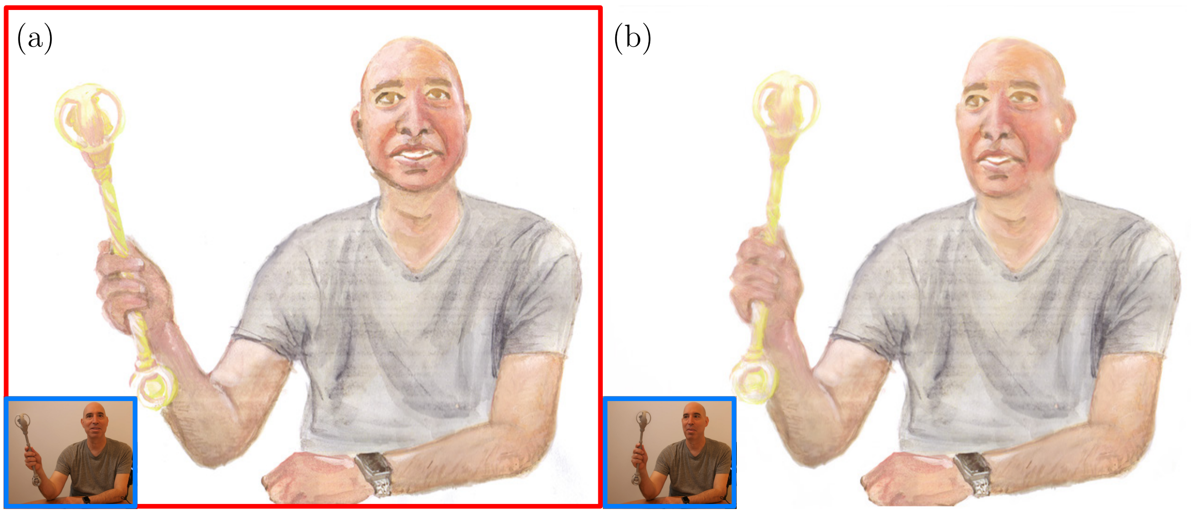


Figure 6.11: *Given one keyframe (a) and a video sequence (in blue), our method produces the stylized result (b). Video frames (insets of a, b) courtesy of © Adam Finkelstein and style exemplars (a) courtesy of © Pavla Sýkorová.*

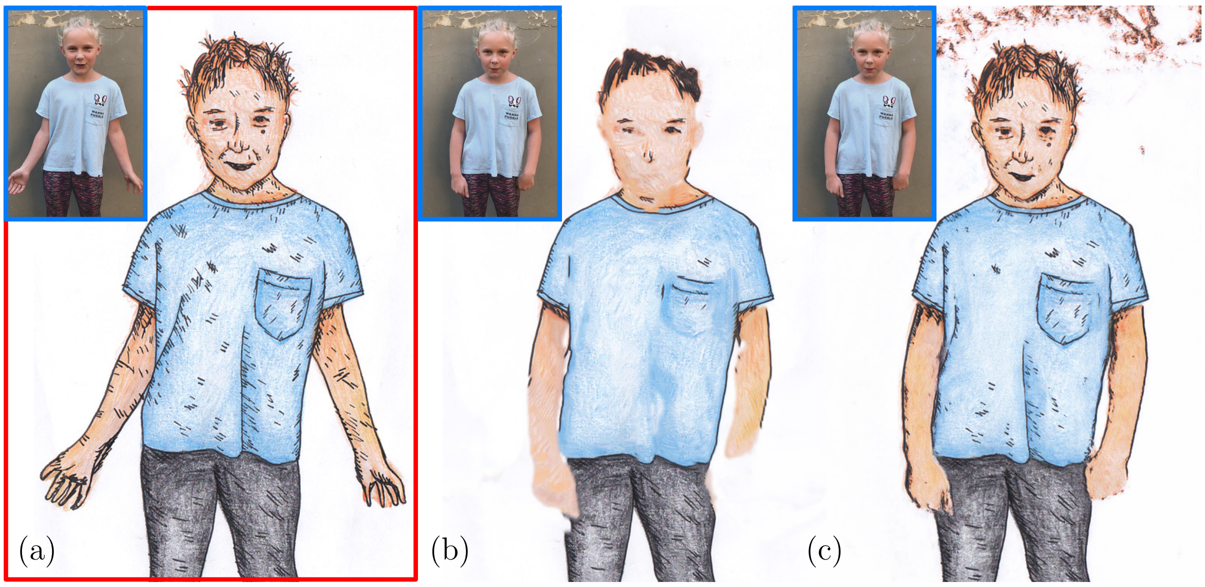


Figure 6.12: *For the state-of-the-art algorithm of [Jam+19], contour based styles (a) present a particular challenge (b). Using our approach (c), the contours are transferred with finer detail and remain sharp even as the sequence undergoes transformations. Video frames (insets of a–c) and style exemplar (a) courtesy of © Štěpánka Sýkorová.*

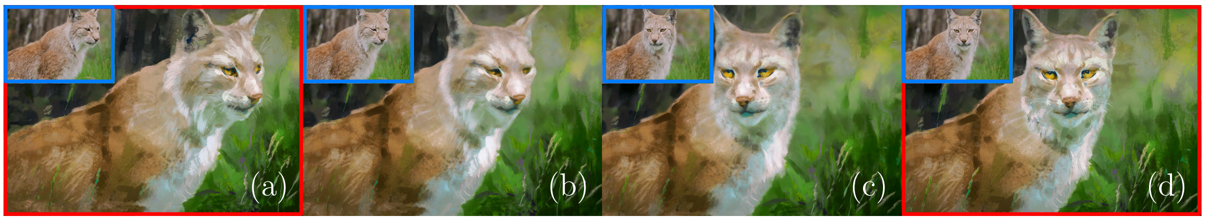


Figure 6.13: *The Lynx sequence stylized using two keyframes (a, d). Notice how our method produces seamless transition between the keyframes while preserving fine texture of the style (b, c). Watch our supplementary video (at 1:22) to see the sequence in motion. Style exemplars (a, d) courtesy of © Jakub Javora.*

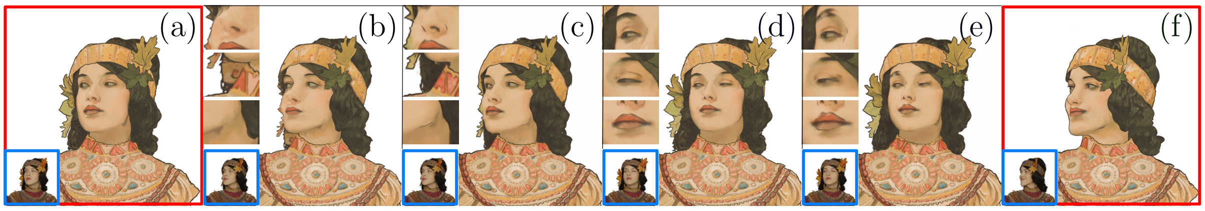


Figure 6.14: *Keyframes (a, f) were used to stylize the sequence of 154 frames. See the qualitative difference between Jamriška et al. [Jam+19] (b) and our result (c). Focusing mainly on zoom-in views, our approach better preserves contour lines around the nose and chin; moreover, the method of Jamriška 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.*

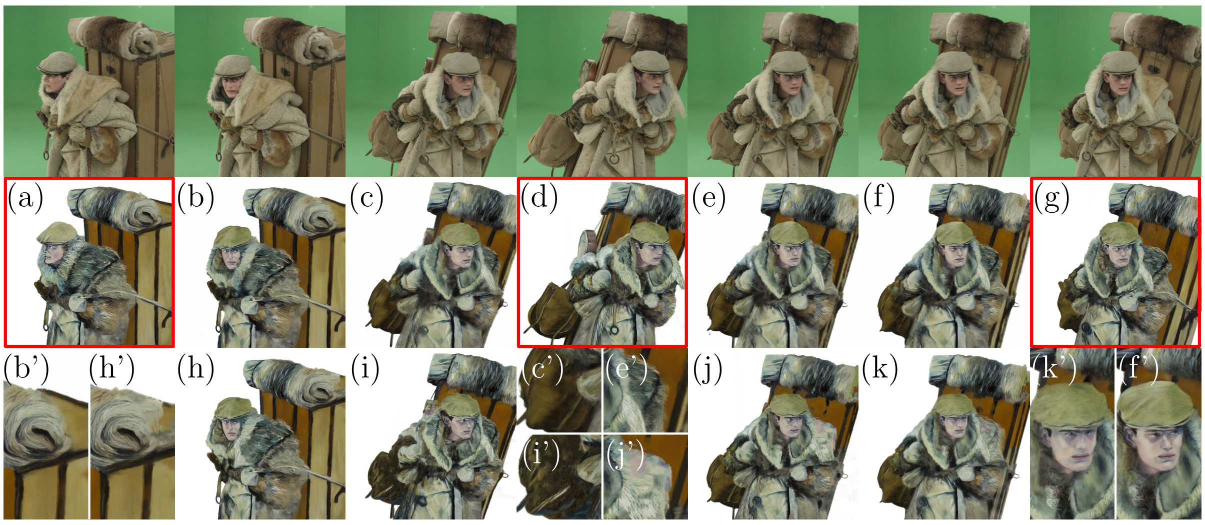


Figure 6.15: *A complex input sequence (the first row) with seven keyframes, three of them are shown in (a, d, g). Here we compare our approach to the approach of Jamriška et al. [Jam+19]. See our result (b) and theirs (h) along with the close-ups (b', h'); due to their explicit handling of temporal 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. [Jam+19] (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.*

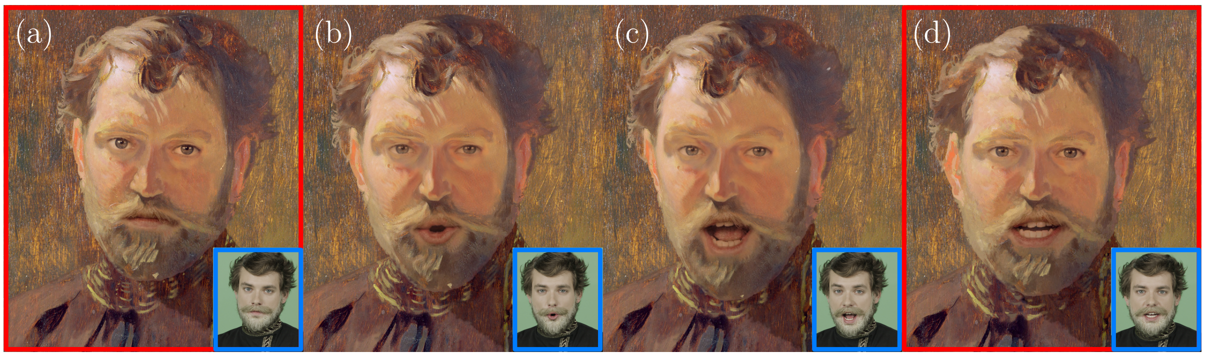


Figure 6.16: *An example sequence of 228 video frames (in blue) as stylized from two keyframes (a, d). Results of our method (b, c) stay true to style exemplars over the course of the sequence. Video frames (insets of a–d) and style exemplars (a, d) courtesy of © Muchalogy.*

Chapter 7

StyleBlit: Stylization with Local Guidance

7.1 Introduction

Example-based artistic style transfer recently became popular thanks to advances made by neural-based approaches [GEB16; SED16], patch-based texture synthesis techniques [Fi16; Fi17] and their combinations [LW16c; Lia+17]. These methods can produce impressive style transfer results with a common limitation of high computational overhead. Although interactive frame-rate can be achieved when compromising visual quality [JAFF16] or utilizing the GPU [Fi16], high-quality style transfer remains out of reach for scenarios such as interactive games or mobile applications where the available computational budget is low.

A key concept that distinguishes style transfer from regular texture synthesis [EL99] is the use of guiding channels [Her+01]. Those encourage the transfer of a specific area in the source exemplar to a corresponding area in the target image. The design of guiding channels is extremely important for achieving semantically meaningful transfer. The guidance can be relatively *fuzzy* with respect to a certain spatial location (e.g., segmentation or blurred gray-scale gradients used by Hertzmann et al.) or well-localized and descriptive (e.g., a displacement field [SED16; Fi17], texture coordinates [Rem+14; Mag+15] or normal values [Slo+01; Dia+15]). We call the latter *local* guidance.

The goal of current state-of-the-art patch-based style-transfer techniques [Fi16; Zho+17] is to optimize for a solution that satisfies the prescribed guidance and consists of large coherent chunks of the style exemplar in semantically meaningful regions. This solution represents the most visually-pleasing configuration that maximizes sharpness and fidelity of the synthesized texture since large areas of the exemplar are copied as is (see Fig. 7.2). To achieve this, however, *textural coherence* [Kwa+05; WSI07] needs to be taken into account which results in a computationally demanding energy minimization problem.

In this paper, we demonstrate that when guidance provide good localization and when style exemplar contains stochastic texture, textural coherence becomes less important as the local characteristics of the guide implicitly encourage coherent solutions and the stochastic nature enables visual masking that suppresses visible seams. In this setting, we demonstrate that expensive optimization can be replaced by a set of simple and fast pixel-level operations that gain significant performance speed-up. On a single core

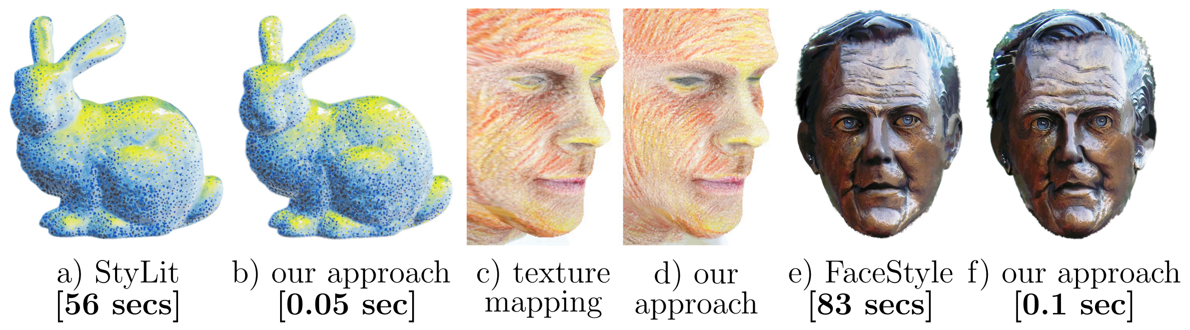


Figure 7.1: *StyleBlit in applications: (a) style transfer from an exemplar in Fig. 7.6 to a 3D model using StyLit [Fi16]; (b) our approach delivers similar visual quality but is several orders of magnitude faster; (c) regular texture mapping using texture presented in Fig. E.8 vs. (d) our approach that better preserves visual characteristics of the used artistic media; (e) style transfer to a portrait image using FaceStyle [Fi17] with an exemplar in their supplementary material; (f) our approach produces similar visual quality and is notably faster.*

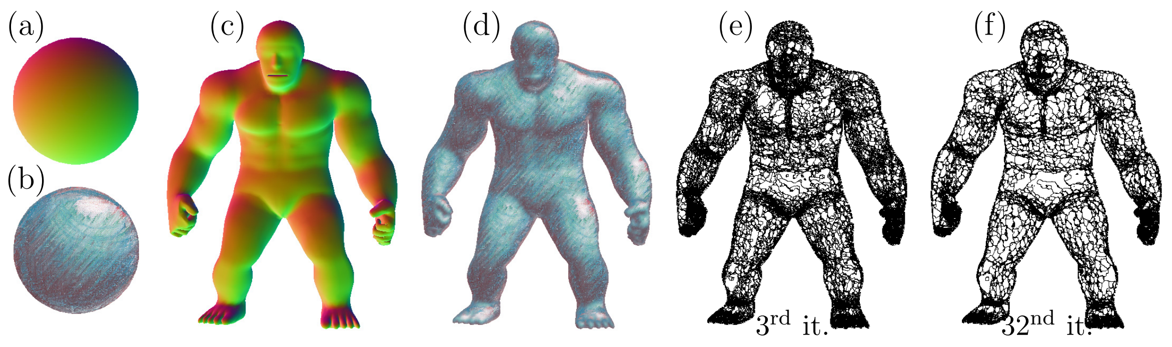


Figure 7.2: *The motivation for our approach: state-of-the-art guided patch-based synthesis [Fi16] is used to transfer artistic style from a hand-drawn sphere (b) onto a more complex 3D object (c). Normal maps are used as guidance (a, c). The result (d) preserves well the textural coherence of the original artistic style exemplar since the optimization-based approach converges to a state where large coherent chunks of the source texture (colored white) are copied into the target image forming a mosaic (e). As the optimization progresses, the size of coherent regions increases (f). Style exemplar:* © Pavla Sýkorová

modern CPU we can stylize a one-megapixel image at 10 frames per second while on a common GPU we can achieve more than 100 frames per second at a 4K UHD resolution. Despite its simplicity, our new method produces high-quality transfer results for a wide range of styles. Applications include stylization of 3D renderings [Fi16] (see Fig. 7.1, left), image-based texture mapping that better preserves the characteristics of natural artistic media [Mag+15] (Fig. 7.1, middle), or fast style transfer to faces with comparable results to the method of Fišer et al. [Fi17] (Fig. 7.1, right). Our technique can also be used in a more generic MatCap scenario [Slo+01] where instead of using explicit shading models a hand-drawn, captured or synthetically prepared photorealistic material is transferred to a more complex 3D object using normal-based guidance (see Fig. 7.9). A key advantage of our approach is that compared to the original solution based on environment mapping [Slo+01] our method transfers larger chunks of the source image, which preserves high-frequency features of the texture.

7.2 Related Work

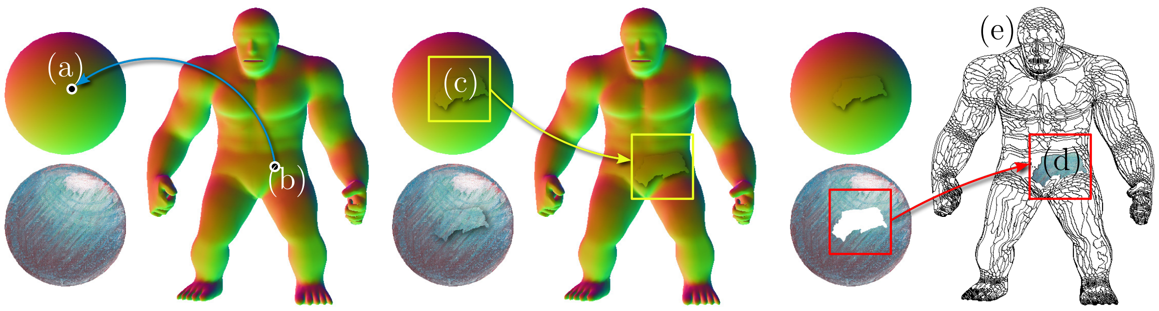


Figure 7.3: *The core idea behind our method: for each randomly selected seed in the target image (b), we perform a table lookup using its guidance value (in this case a normal) to retrieve the corresponding location in the source exemplar (a). Then we compare the guidance values of source and target pixels in spatially-aligned regions around the seed. Pixels with a guidance value difference below a user-defined threshold belong to the same chunk (c). Finally, we transfer the chunk of example pixels to the target (d). We can produce the final mosaic by repeating this process (e). Style exemplar:* © Pavla Sýkorová

Over the last two decades, non-photorealistic rendering [Kyp+13] evolved considerably. The state-of-the-art techniques can synthesize images resembling real artwork. A popular branch of techniques achieves this goal by mixing a set of predefined strokes or patterns that are selected and positioned according to guiding information provided in 2D [Her98] or 3D [Sch+11] environments. In addition to painterly styles, this line of approaches can also simulate other artistic styles such as pen-and-ink illustration [Sal+97] or hatching [Bre+07]. Nevertheless, these approaches are confined by the limited expressive power of these predefined sets of strokes or patterns.

To alleviate this drawback, an example-based approach called *Image Analogies* was introduced by Hertzmann et al. [Her+01]. This method allows an artist to prepare an arbitrary stylized version of a target image given an input style example. A one-to-one mapping between the input image and its stylized version is used to guide the transfer by establishing correspondences between the source and target (based, e.g., on color correspondence). The target image can then be stylized according to this analogy. This seminal concept was later extended to animations [Bén+13] and improved by others [BZ17] using better synthesis algorithms [Kas+15; Fi16] as well as different types of guidance [Zho+17; Fi17]. In parallel, an approach similar to *Image Analogies* was introduced by Sloan et al. [Slo+01] and later extended by others [BTM06; TAY13]. Their technique called *The Lit Sphere* (a.k.a. *MatCap*) uses a one-to-one correspondence between normal values to transfer style from a hand-drawn exemplar of a simple object (a sphere) to a more complex 3D model. In this scenario, a simple environment mapping can be used [BN76] to perform the transfer. Recently, Magnenat et al. [Mag+15] proposed a similar technique where instead of normals, UV coordinates are used as guidance so that the artist can draw a stylized version on a 2D projection of a 3D model and then the style is transferred using texture mapping. This approach is similar to image-based texture mapping used in 3D reconstruction [DTM96]. Style transfer can be performed in real-time thanks to its simplicity, but it only works well when the style does not contain distinct high-frequency details. Texture mapping often distorts high-frequency details failing to retain the fidelity of the used artistic medium. Later patch-based synthesis methods [Fi16; BKR17] have obtained much higher quality results by taking into account not only local guidance but also textural coherence. These improvements, however, came at the cost of notably higher computational overhead.

Recently, Gatys et al. [GEB16] introduced an alternative approach to style transfer based on parametric texture synthesis [PS00] where instead of a steerable pyramid, an alternative parametric representation is used based on a deep neural network trained for object recognition [SZ14]. Their technique inspired a lot of follow-up work [SID17] and became very popular thanks to numerous publicly available implementations. Although it produces impressive results for some style exemplars, it was shown to suffer from certain high-frequency artifacts caused by the parametric nature of the synthesis algorithm [Fi16; Fi17]. To prevent texture distortion, researchers have proposed techniques to combine the advantages of patch-based synthesis and the deep features learned by neural network [LW16c; Lia+17]. These approaches, however, have significant computational overhead and are not suitable for real-time applications.

Our approach to style transfer bears resemblance to early texture synthesis approaches [PFH00; Lia+01; EF01; Kwa+03] that can achieve results similar to patch-based synthesis [Kwa+05; WSI07] by transferring larger irregularly-shaped chunks of the source exemplar and composing them seamlessly in the target image. In particular *Lapped Textures* [PFH00] can tile the target surface with a set of source patches, however, there is no specific guidance for the patch placement, the patches need to be prepared in advance to have minimal features on boundaries (to avoid seams), and the approach requires an additional growing operation to fill in gaps. In appearance-space texture synthesis [LH06], small appearance vectors are used instead of color patches to compress neighborhood information, but an iterative optimization [LH05] is still necessary to obtain the final result.

In another related work [PKVP09], a graph labeling problem is solved to find the optimal shift of every pixel in the output image from its source in an input image. Nevertheless, additional smoothness term is needed to avoid discontinuities, and so computationally demanding optimization is required.

In this paper, we demonstrate that for style exemplars which contain mostly stochastic textures the interplay between local guidance and textural masking effect described by Ashikhmin [Ash01] makes seams between the individual chunks barely visible and thus simple blending operation can be used to suppress them without the need to take into account texture coherence explicitly.

7.3 Our Approach

In this section, we describe the core idea behind our approach and discuss implementation details. As a motivation, we first describe a simple experiment that inspired us to develop our method.

To understand the properties of optimization-based approaches, we applied the StyLit algorithm [Fi16] to transfer the style from a hand-drawn image of a sphere to a more complex 3D model using normals as guidance (see Fig. 7.2). The texture coherence term in the original energy formulation, and the mechanism for preventing excessive utilization of source patches, help the optimization converge to a state where large chunks of the original source texture (Fig. 7.2b) are copied to the target image resulting in a high-fidelity transfer (Fig. 7.2d).

Inside each coherent chunks, the errors of texture coherence term are equal to zero. Errors of the guidance term can be bounded by a small upper bound, i.e., we can find a chunk of the normal field on the exemplar sphere to roughly approximate the corresponding chunk of normals on the target 3D model within a certain error threshold. The black lines in Fig. 7.2e, f show the boundaries between chunks within which all pixels have guidance errors below some predefined error bound. The lines get sparser and the regions grow larger as the bound increases.

This fact inspired us to seek large coherent chunks of style regions directly using simple pixel-level operations foregoing expensive patch-based optimization.

7.3.1 Basic Algorithm

To build such a mosaic of coherent chunks, we need to estimate the shape and spatial location of each individual chunk. This is done by going in the scan-line order or by picking a random pixel (seed) in the target image and finding its corresponding location in the source exemplar (see Fig. 7.3a, b). Usually, the local guidance at each target pixel consists of two values that indirectly specify the corresponding pixel coordinates in the source exemplar. This fact enables us to use a simple look-up table to retrieve, for each target pixel, the corresponding location in the source exemplar. In a more complex scenario where additional guiding channels are used, we can accelerate the retrieval using search trees [Ary+98]. Once we know the corresponding source pixel, we calculate the difference between the guidance values in local spatially-aligned regions. The target pixels having guidance difference smaller than a user-defined threshold belong to the current chunk (Fig. 7.3c). We copy those corresponding pixels and paste them in the target image (Fig. 7.3d). By repeating the searching and copying steps, we eventually cover all pixels in the target image (Fig. 7.3e and Fig. 7.7, left).

Our approach does not explicitly enforce textural coherence. One might expect that seams between individual chunks will be visible. Surprisingly, for a relatively large variety of exemplars, seams are either not apparent or can be effectively suppressed using linear blending applied around the boundaries of individual chunks. The reasons are twofold: (1) local guidance is often smooth and continuous and thus two neighboring chunks are usually roughly aligned; (2) hand-drawn exemplars are typically highly stochastic which intrigues the human visual system and makes the structural inconsistencies less noticeable [Ash01].

7.3.2 Implementation Details

The basic algorithm can be implemented in a brute-force manner (see supplementary material for pseudocode). Though simple, it is highly inefficient due to the redundant visiting of target pixels and the inherent sequential nature that prohibits parallel implementation.

To overcome the mentioned drawbacks, we use a more efficient approach that is fully parallel and guarantees that every target pixel will be visited only once (see Algorithm 1). The key idea here is to define an implicit hierarchy of target seeds $\boldsymbol{q}$ (see Fig. 7.4) with different granularity. On the top level, seeds are distributed randomly far apart. On

Algorithm 1: ParallelStyleBlit

Inputs : target pixel $\boldsymbol{p}$, target guides G_T, source guides G_S, source style exemplar C_S, threshold t, number of levels L.

Output: stylized target pixel color $C_T[\boldsymbol{p}]$.

```
SeedPoint(pixel $\boldsymbol{p}$, seed spacing $h$):
    $\boldsymbol{b} = \lfloor \boldsymbol{p}/h \rfloor$; $\boldsymbol{j}$ = RandomJitterTable[$\boldsymbol{b}$]
    return $\lfloor h \cdot (\boldsymbol{b} + \boldsymbol{j}) \rfloor$

NearestSeed(pixel $\boldsymbol{p}$, seed spacing $h$):
    $d^\star = \infty$
    for $x \in \{-1, 0, +1\}$ do
    for $y \in \{-1, 0, +1\}$ do
        $\boldsymbol{s}$ = SeedPoint($\boldsymbol{p} + h \cdot (x, y)$, $h$)
        $d = ||\boldsymbol{s} - \boldsymbol{p}||$
        if $d < d^\star$ then
            $\boldsymbol{s}^\star = \boldsymbol{s}$; $d^\star = d$
    return $\boldsymbol{s}^\star$

ParallelStyleBlit(pixel $\boldsymbol{p}$):
    for each level $l \in (L, \ldots, 1)$ do
        $\boldsymbol{q}_l$ = NearestSeed($\boldsymbol{p}$, $2^l$)
        $\boldsymbol{u}^\star = argmin_{\boldsymbol{u}} ||G_T[\boldsymbol{q}_l] - G_S[\boldsymbol{u}]||$           ← found via lookup,
        $e = ||G_T[\boldsymbol{p}] - G_S[\boldsymbol{u}^\star + (\boldsymbol{p} - \boldsymbol{q}_l)]||$           or a tree search.
        if $e < t$ then
            $C_T[\boldsymbol{p}] = C_S[\boldsymbol{u}^\star + (\boldsymbol{p} - \boldsymbol{q}_l)]$
            break
```

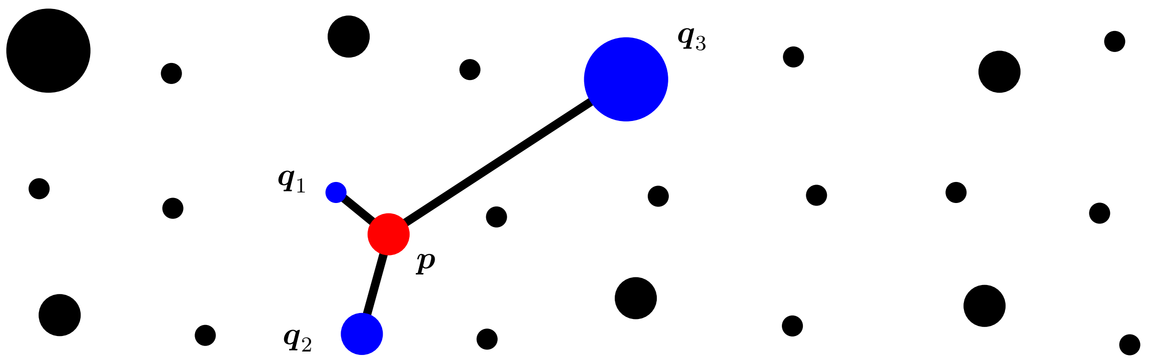


Figure 7.4: *An example hierarchy of spatially distributed seeds* $\boldsymbol{q}_l$ *(black and blue dots). The hierarchy level* l *corresponds to the size of the dots: the dots in the top level are the largest. For every target pixel* $\mathbf{p}$ *(red dot), we proceed from the top level to the bottom* $l = \{3, 2, 1\}$*. At the top level, we retrieve the spatially nearest seed* q_3*, and check whether the guidance value between* p *and* q_3 *falls below a specified threshold. If not, we proceed to the nearest seed in the next lower level* q_2 *and then* q_1*.*

the lower levels, the distance between them is gradually decreased by a factor of 2. Algorithmically we build this hierarchy by placing dots at regular grid points whose positions are randomly perturbed. Then for every target pixel $\boldsymbol{p}$, we start at the top level of our seed hierarchy and find the spatially nearest target seed $\boldsymbol{q}_l$ within the same level l.

If the nearest seed yields guidance error below a specific threshold, we transfer the corresponding style color to the target pixel and stop the traversal, otherwise we enter the next lower level of the hierarchy and continue until we reach the bottom level.

When seams become apparent, we can optionally perform blending on the boundaries of individual chunks. This can be simply implemented by replacing the transfer of pixel colors with the transfer of pixel coordinates, i.e., every target pixel will be assigned its corresponding source pixel coordinates. This structure is equivalent to the nearest neighbor field used in patch-based synthesis. Then, the final colors are obtained using a voting step [Kwa+05; WSI07] where the color of every target pixel is computed as the average color of co-located pixels from a set of source patches that intersect the currently processed target pixel. This operation is simple to implement and is, in fact, equivalent to performing blending only at chunk boundaries.

7.3.3 Extensions

Our method is suitable both for hand-drawn style exemplars as well as realistic materials that have stochastic nature. Those, however, may contain smooth gradients together with high-frequency features (see Fig. 7.5a). In this case, finding a threshold that would preserve both smoothness and high-frequency details could be difficult (Fig. 7.5b). We resolve this problem by employing a multi-layer approach [BA83; Han+08; Gui+17]. We first separate the input style exemplar into a smooth base layer (Fig. 7.5c) and a high-frequency detail layer (Fig. 7.5d). To obtain the base layer, we first filter the original style image with Gaussian filter and then we subtract the filtered image from the original to get the detail layer. Style transfer is then performed in each layer separately. In the base

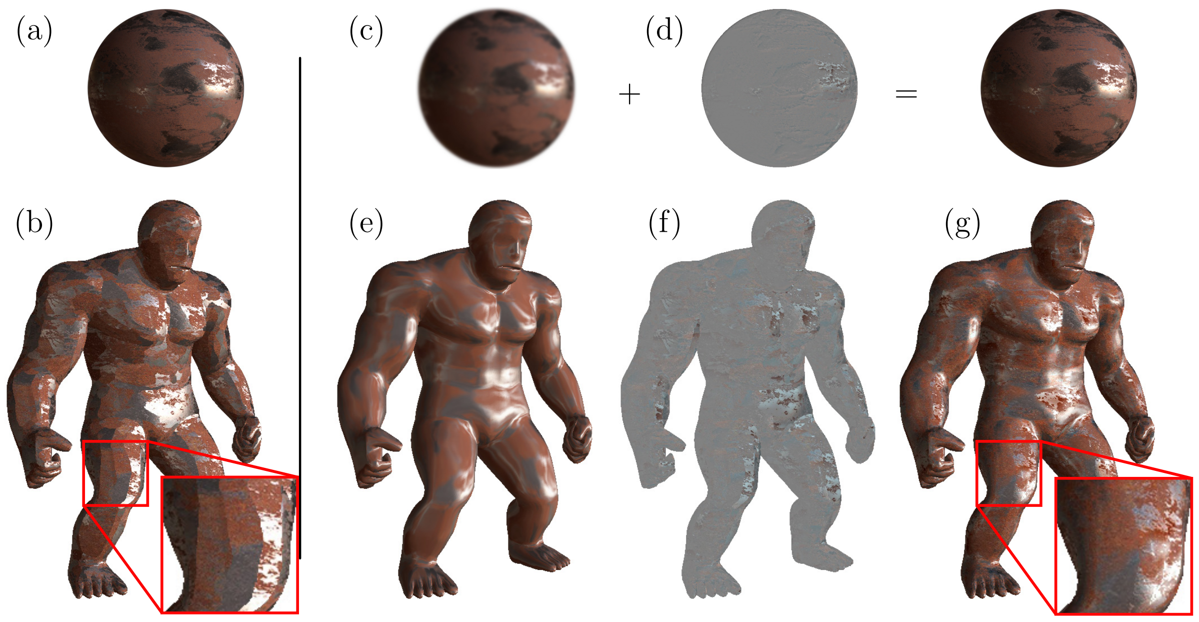


Figure 7.5: *Multi-layer approach: style exemplar with smooth gradients and high-frequency details (a) may introduce visible seams (b). By decomposing the exemplar into base (c) and detail (d) layer one can employ The Lit Sphere algorithm [Slo+01] for the base (e), then apply our algorithm on the detail (f), and finally, make the composition which preserves both smoothness as well as high-frequency details (g). Style exemplar:* © Free PBR

layer, we employ The Lit Sphere algorithm [Slo+01] which works well for low-frequency content (Fig. 7.5e). For the detail layer, we apply our algorithm which preserves high-frequency content (Fig. 7.5f) and finally, we make the seamless composition by summing synthesized base and detail layers (Fig. 7.5g).

Our approach can also be extended to animations. The local guidance implicitly encourages temporal coherence in the synthesized content while the randomization of seed points slightly perturbs the structure of the resulting mosaic. This creates a slight temporal flickering effect which gives the observer an illusion of a hand-colored animation where every frame is drawn independently by hand [Fi14]. Moreover, the amount of flickering can be controlled by changing the guidance threshold. Higher threshold gives rise to larger chunks and more visible visual changes between consecutive frames, and thus the amount of flickering is increased.

7.4 Results

We implemented our approach on the CPU using C++ and on the GPU using OpenGL with GLSL (for desktop) as well as WebGL (for mobile devices). As a default threshold value, we use $t = 24$ and the number of seed levels is set to $L = 7$. Table `RandomJitterTable` contains random values between $(0, 1)$. On a single core CPU (Core i7, 2.8 GHz), we stylize a one-megapixel image at 10 frames per second while on the GPU (GeForce GTX 970) we can achieve more than 100 frames per second at 4K resolution. This represents three orders of magnitude speedup as compared to the original StyLit algorithm [Fi16] which requires computationally demanding iterative optimization. Such improvement enables us to perform real-time style transfer even on devices with a lower

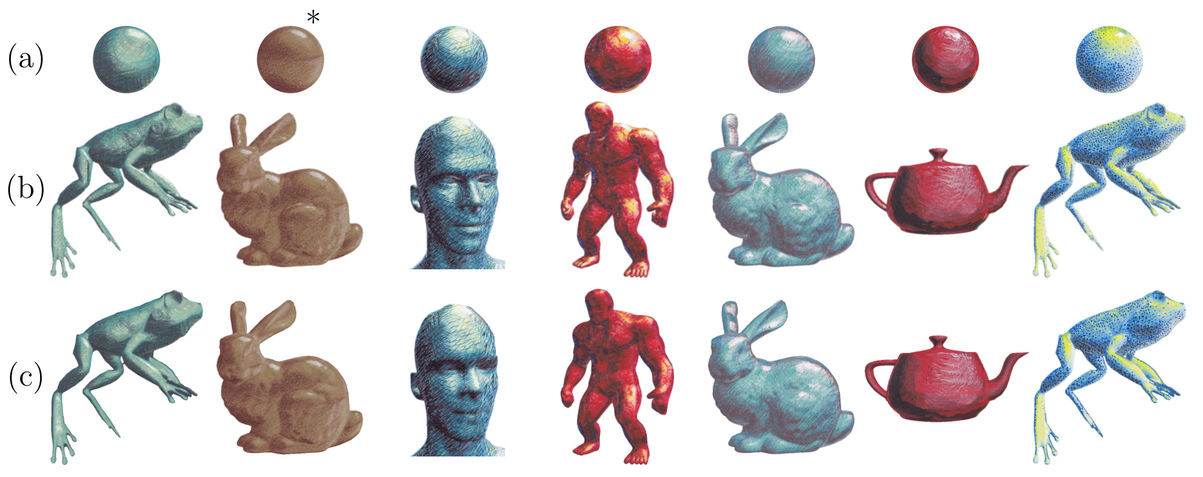


Figure 7.6: *Comparison with StyLit [Fi16]: original style exemplar (a), the result of our approach (b), and the result of StyLit (c). Style exemplars:* © Pavla Sýkorová *and* Daichi Ito*

computational budget including mid-range mobile phones (using WebGL 1.0 we can achieve, e.g., 15 frames per second full screen on the Samsung Galaxy A3).

We tested our approach in three different style-transfer scenarios where local guidance is used: normals (see Fig. 7.6 and 7.9), texture coordinates (Fig. 7.7 and E.8), and a displacement field (Fig. E.9). For additional results see also Fig. 7.1 and the supplementary material.

For normal-based guidance, we compared our approach with the StyLit algorithm [Fi16] to confirm that we produce comparable results that preserve visually important characteristics of artistic media (see Fig. 7.1, 7.7, 7.6, and the supplementary material that includes results of a perceptual study). In addition, our approach also better preserves geometric details (cf., e.g., head result in Fig. 7.6) since it compares guidance channels per pixel and does not involve any patch-based averaging used in the StyLit algorithm. Such averaging acts as a low-pass filter applied on the guidance channel. In the supplementary video, we present a recording of an interactive session (on the GPU as well as on a smartphone) where the user manipulates and animates a 3D model on which a selected artistic style is transferred in real-time. We also demonstrate controllable temporal flickering effect following the concept of Fišer et al. [Fi14]. Our approach is suitable also for transferring delicate pixel art styles where even small blurring artifacts may become apparent (see Fig. 7.8).

We also compare our technique with The Lit Sphere algorithm [Slo+01], i.e., MatCap scenario which is based on environment mapping. It directly maps colors between corresponding pixels according to a one-to-one mapping specified by the normal values. Due to pixel-level processing, high-level structures visible in the style exemplar become distorted, and thus only low-frequency exemplars can be used. In contrast, our approach copies larger chunks and thus better preserves high-level structures which are important to retain fidelity of the original style exemplar (see Fig. 7.7, Fig. 7.9 and the supplementary material). This improvement is visible also in the case where texture coordinates are derived directly from a planar parametrization (unwrap) of the target 3D mesh (see Fig. 7.1, E.8, and the supplementary material). Here the style exemplar can be painted on a specific 2D projection of the 3D mesh [Mag+15] or directly on the planar unwrap. In both cases, our approach transfers larger chunks of the original texture

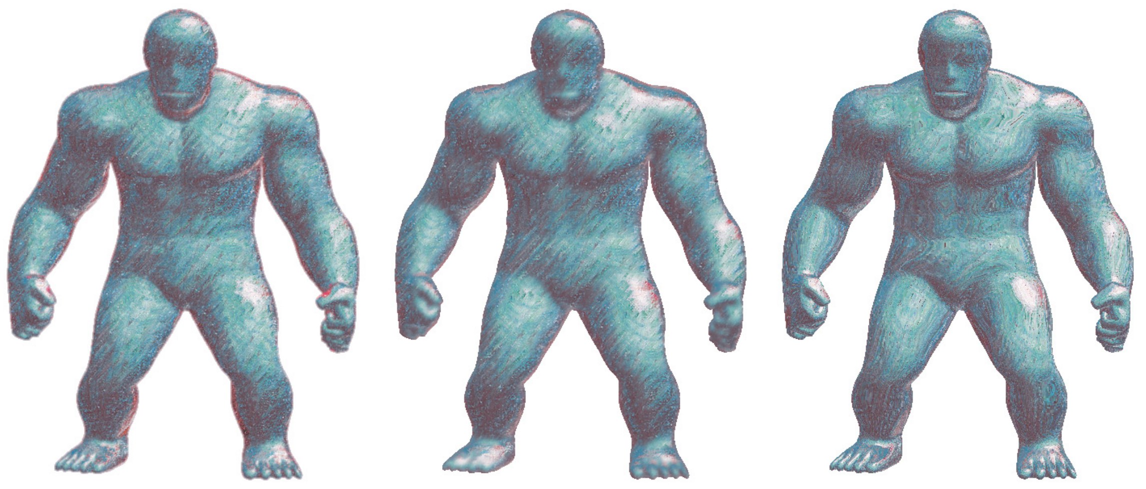


Figure 7.7: *Stylized results produced by our method (left), StyLit [Fi16] (middle) and The Lit Sphere [Slo+01] (right). Compared to StyLit, our approach is orders of magnitude faster and produces similar result quality without explicitly enforcing textural coherence. Compared to The Lit Sphere, our algorithm is equally fast, but retains the high-level structure of the used artistic media; i.e., large directional brush strokes are better preserved.*

which effectively removes artifacts caused by texture mapping and better preserves the fidelity of the style exemplar. To do that, however, a larger threshold is required which can break the structure of high-level geometric features. To avoid this artifact, we use additional segmentation guide which prevents chunks from crossing boundaries of semantically important regions (see supplementary material for examples of these additional guiding channels).

Finally, we tested our approach in a scenario where a dense displacement field is used as a local guide. An example of such setting is artistic style transfer to human portraits [Fi17]. Here the displacement field is defined by a set of corresponding facial landmarks detected in the source exemplar and in the target subject. Moving least squares deformation [SMW06] is used to compute dense correspondences, i.e., the resulting displacement field. Besides the local guide, two additional guidance channels are used for patch-based synthesis: a segmentation map containing semantically important facial parts (head, hair, eyes, eyebrows, nose, and mouth) and an appearance guide that helps to preserve subject's identity (see the supplementary material for examples of all guiding channels). The resulting visual quality is comparable or a bit inferior to the previous work, but sufficient for applications with limited computational resources (see Fig. 7.1, E.9, and the supplementary material). To demonstrate such an application a recording of a live session with real-time facial style transfer to a video stream is presented in the supplementary material. To highlight the benefit of our method, the result of our algorithm is compared side-by-side with a simple texture mapping scheme. Note how our approach better preserves the fidelity of the original artistic media.

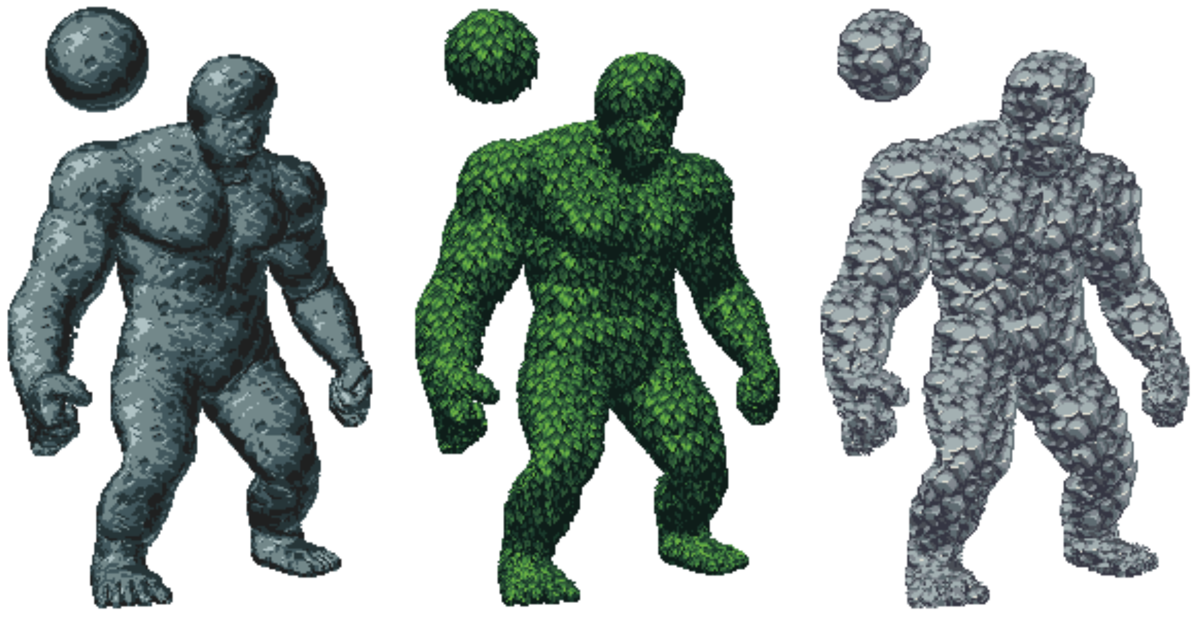


Figure 7.8: *Examples of stylization where normal-based guidance is used to transfer delicate pixel art styles. In this scenario, copy-and-paste nature of our approach is crucial as it allows to retain essential details on the pixel level which are important to preserve the fidelity of images that has been created manually pixel by pixel. Style exemplars: © Lachlan Cartland*

7.5 Limitations and Future Work

Although our method produces visually pleasing results for a variety of different style exemplars and different types of guidance, there are some limitations that need to be taken into account.

For non-stochastic (semi-)regular textures like a brick wall, our approach may introduce visible misalignment of regular structures (see Fig. 7.12a). To suppress this artifact one may employ post-transfer alignment of individual chunks using the method of Lucas and Kanade [LK81]. This operation can be performed relatively quickly as it requires only inexpensive accumulation of image gradients and pixel differences over chunk boundaries and since the misalignment is usually small, only a few iterations are necessary to get a better alignment (see Fig. 7.12b). Nevertheless, the quality is still inferior as compared to full-fledged synthesis (see Fig. 7.12c).

Visible misalignment of individual chunks can also be apparent in cases when a set of guidance channels used for the style transfer does not contain local guide or when the influence of local guide is low as compared to other channels. Example of such scenario can be the usage of light path expressions in [Fi16] (see Fig. 7.13a). In this case, we envision a more sophisticated post-transfer alignment mechanism would also handle larger discrepancies.

Our approach shares limitations with techniques that use guided patch-based synthesis [Kas+15; Fi16]. They may produce excessive repetition in cases when the scale of the target object is fairly different as compared to the object in the style exemplar, e.g., during zoom-in operations or when there is not enough variability in the guidance, e.g., when stylizing flat surface using spherical exemplar (see Fig. 7.13b). This drawback can be alleviated by adjusting the global scale or by preparing a different style exemplar that contains similar structures as the target objects.

Another limitation is related to the rotation in the image plane when texture coordinates or displacement field are used for guidance. In this situation corresponding counterparts of target seeds can be found easily, however, as their neighborhoods have notably different content caused by rotation, the error threshold limits the size of the target chunks, and the method will introduce blur into the result (see Fig. 7.13c). To alleviate this issue, one can pre-rotate the source guidance to match with the dominant orientation in the target channel as in [Fi17].

7.6 Conclusion

We have presented a new approach for example-based style transfer suitable for applications where strong local guidance is used. We demonstrated that in this scenario computationally demanding patch-based synthesis converges to a solution that can be easily mimicked using a relatively simple algorithm with notably lower computational overhead. We also showed that considering textural coherence is not crucial for successful style transfer as local guidance in conjunction with the visual masking effectively suppresses visible seams for a variety of hand-drawn as well as photorealistic style exemplars. Since our method is several orders of magnitude faster as compared to the current state-of-the-art, it enables real-time style transfer even in applications with limited computational resources available.

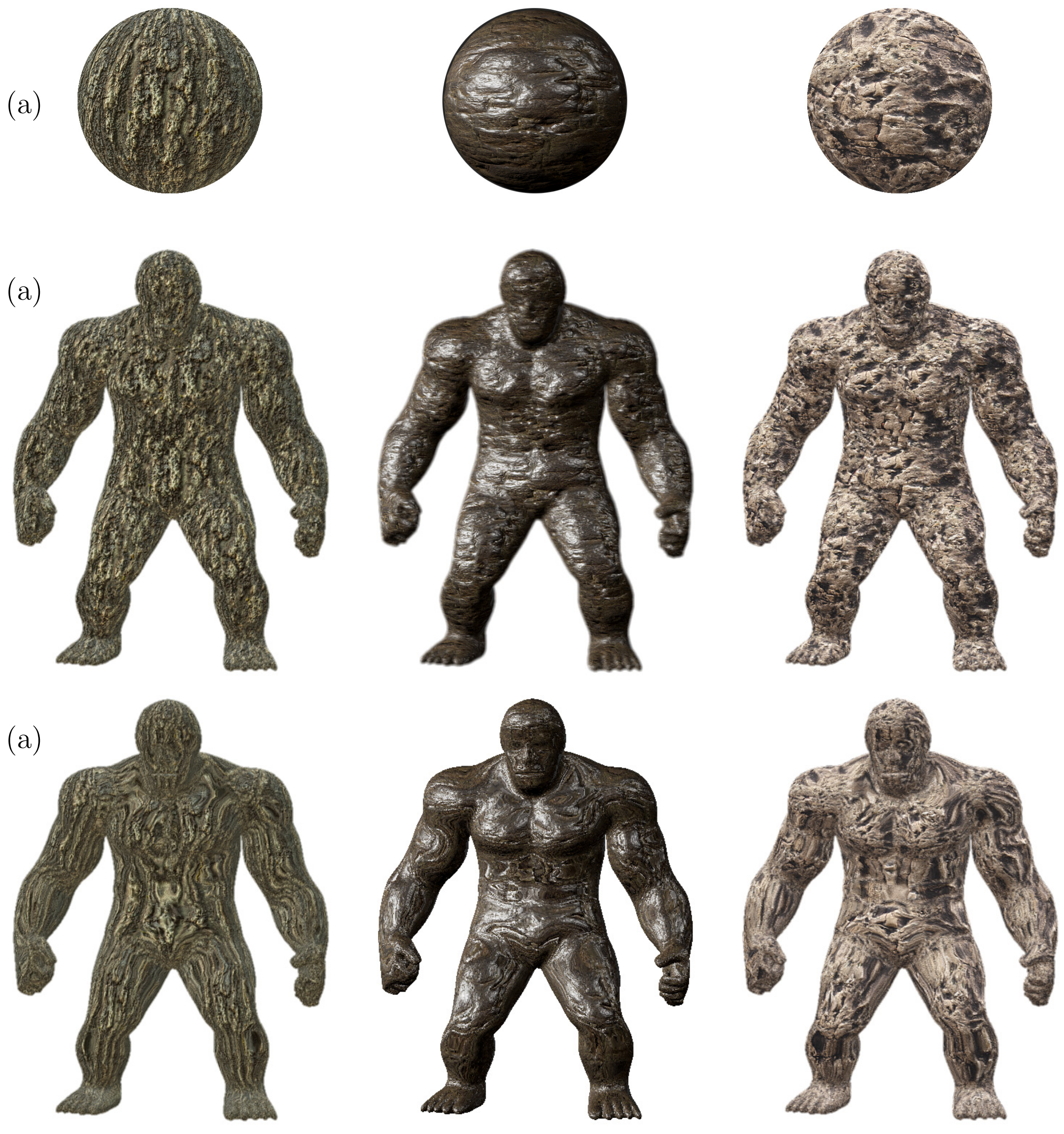


Figure 7.9: *Comparison with The Lit Sphere [Slo+01]: style exemplar (a), our approach (normal-based guidance) (b), and The Lit Sphere result (c). Note how our approach enables MatCap scenario also for materials that contain distinct high-level features while the computational overhead is still comparable to the original Lit Sphere method which is not applicable in this context. Style exemplars:* © Free PBR

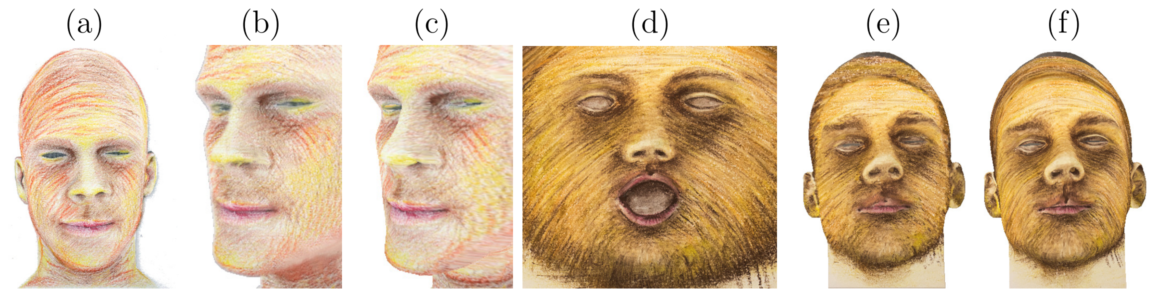


Figure 7.10: *Comparison with texture mapping: original artwork (a, d), new viewpoint generated using our approach (b, e) and using texture mapping (c, f). Style exemplars:* © Pavla Sýkorová

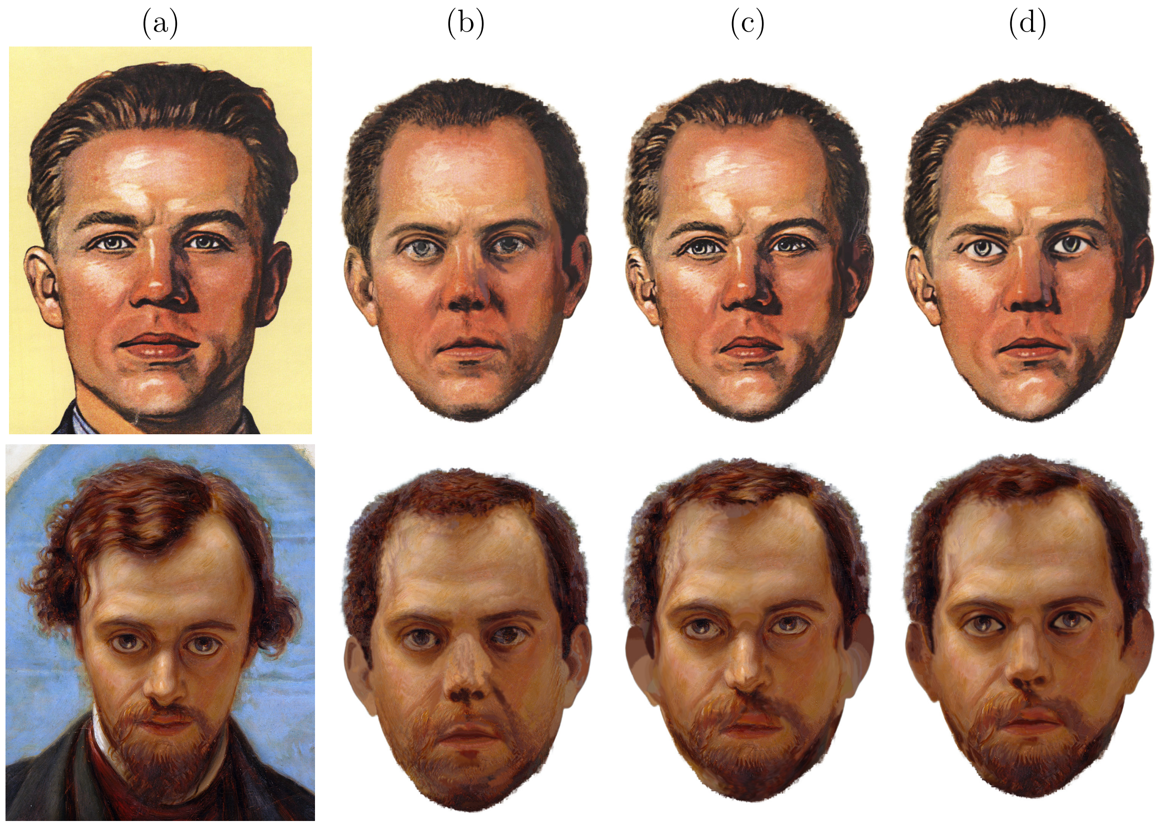


Figure 7.11: *Comparison with FaceStyle [Fi17]: original style exemplar (a), the result of our method using strong (b) and weak (c) appearance guide, and the result of FaceStyle (d).*

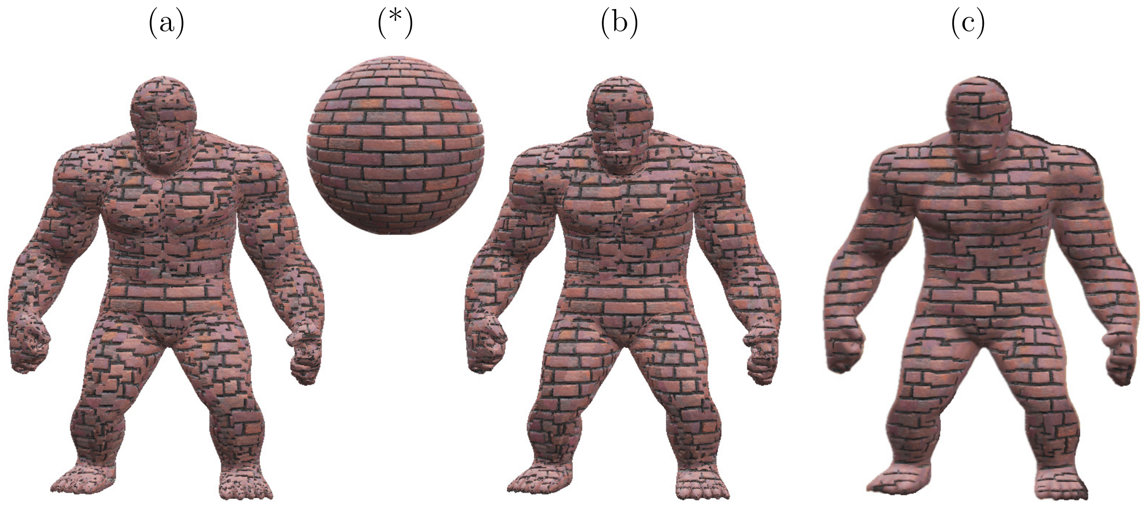


Figure 7.12: *Limitation: when a (semi-)regular texture (*) is used as a style exemplar, our method may introduce visible misalignment of regular features (a). To suppress this artifact, post-transfer alignment of individual chunks can be performed (b) to get a result which is closer to the output of StyLit algorithm [Fi16] (c). Style exemplar:* © Free PBR

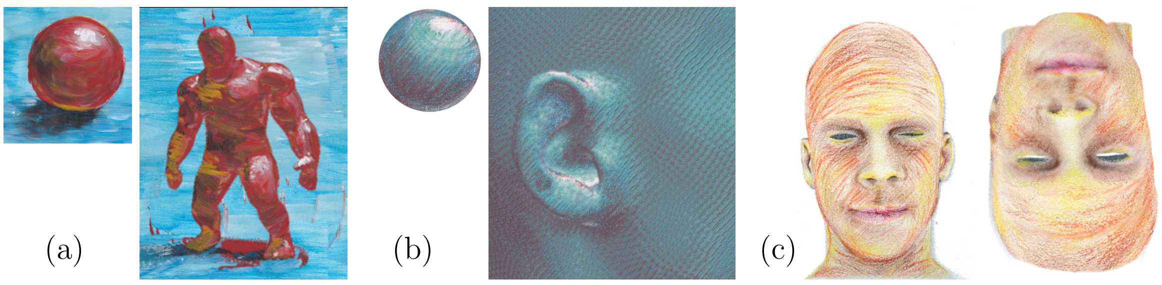


Figure 7.13: *Limitations: when a set of guiding channels does not contain local guide, for instance when light path expressions are used [Fi16], our approach may introduce visible seams (a); when the target contains large areas of pixels having constant guidance values, our method produces a visible texture repetition (b); when the orientation of local guide changes considerably (vertically flipped), translation cannot accommodate this change, and our technique starts to produce smaller chunks (c). Style exemplars:* © Pavla Sýkorová

Chapter 8

StyleProp: Stylization of 3D Models

8.1 Introduction

With the rapid evolution of physically-based rendering and the ability to reproduce natural materials' appearance, artists nowadays produce breathtaking animated movies and video games that are quickly reaching a state of absolute visual perfection. Although the audience highly appreciates this convergence, artists start to feel that with the prevalence of realism, the visuals they continue producing become less and less unique. It is usually challenging for an uninformed observer to recognize an animated movie's authorship or a video game by its visuals. Due to this reason, artists start to seek techniques that can automatize repetitive tasks while still being able to retain their unique style. The ability to draw by hand either physically or digitally and reproduce the look of traditional artistic media has recently become increasingly attractive (see, e.g., games such as *Cuphead*, *Memories Retold*, *Dreams*, *Machinarium*, or *Dordogne* and recently released animated features *Spider-Man: Into the Spider-Verse* and *Loving Vincent*, Disney shorts *Just A Thought* and *Jing Hua* or Riot Games' *Annie*).

Besides the production of animated movies and video games, a similar trend also emerges in other fields where visual uniqueness plays an important role. For instance, in the architecture design, when a studio participates in a competition to realize a devel-

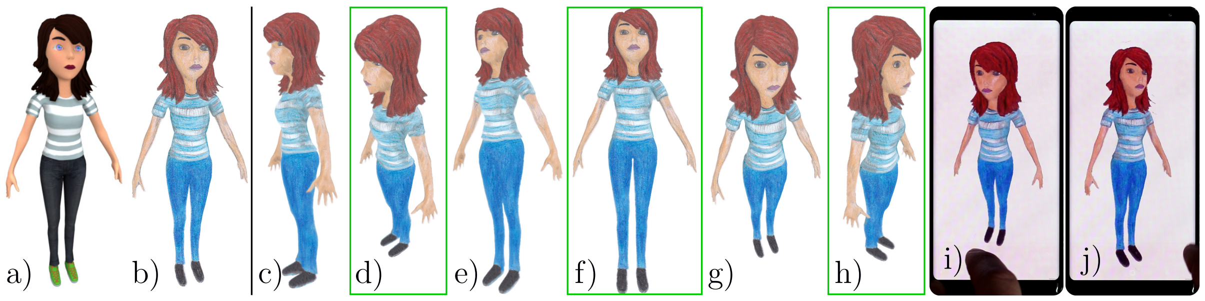


Figure 8.1: *StyleProp in action: a hand-drawn style is transferred to a given 3D model (a) from a single exemplar created using color pencils (b). A novel variant of guided patch-based synthesis is used to pre-calculate a sparse set of samples (d, f, h) from which the model can be rendered in real-time at arbitrary location within available interaction space (c, e, g) even on a mobile phone (i, j) while maintaining consistency when the viewing direction is changed. Style exemplar (b) courtesy of © Štěpánka Sýkorová.*

oper project, photo-realistic visualizations are usually considered a disadvantage. They prevent the committee from recognizing the unique style of a particular studio, which indirectly serves as a quality certificate. Nevertheless, creating such a distinctive presentation is a tedious task; thus, there is a high demand for tools that could help automatize the creative process while still retaining the original aesthetic quality.

An ideal tool that would help artists to simplify the creative process in the sense mentioned above would take a stylized example (e.g., an initial view on a 3D model), distill its distinct visual properties, and transfer them on the target content (e.g., the same model in different viewpoint or pose) so that the resulting stylized counterpart reproduces the look and the feel of the original artwork.

Such a setting is in line with the current research efforts on automatic style transfer that became popular thanks to significant advances made by neural techniques [GEB16; KSS19; Kot+19a]. Despite the impressive results those approaches can produce, their fundamental limitation is that they are trying to reproduce only the given artistic style's statistical properties. There is no guarantee that a specific local stylization choice made by an artist (e.g., a carefully crafted stroke depicting an eye region) will retain in the stylized counterpart.

A concurrent approach to neural style transfer uses guided patch-based synthesis [Her+01; Bén+13; Fi16; Fi17; Jam+19], which focuses more on local textural details and semantic meaningfulness of the transferred style instead of global statistics. By taking into account those essential properties, the results produced by those techniques are sometimes difficult to distinguish from the original artwork. However, their drawback is a significant computational overhead that hinders their applicability in interactive applications. Although real-time approximative solutions exist [Fut+19; Sýk+19], those impose various restrictions on the content being stylized (e.g., faces only) and the type of guidance that can be used (e.g., sufficient spatial variation).

In this paper, we introduce a novel solution to guided patch-based synthesis that enables real-time response with temporal coherence while being agnostic to the stylized content and guidance. We sparsely sample the space of possible interaction states (e.g., camera rotations) and compute each state's stylization coherently with nearby samples. Then for each stylized state, we store only its latent representation (the nearest-neighbor field) from which we can quickly reconstruct the intermediate states and render the final stylized image. Moreover, since the sampled set is relatively compact, we can transfer it swiftly via the network and deliver a smooth interactive 3D viewing experience even on a mobile device.

8.2 Related Work

Early approaches to non-photorealistic rendering [Kyp+13] use hand-crafted algorithmic solutions to paint an input image or video in a particular style. Some employ physical simulation [Cur+97; Hae+07; LXJ12] or a hand-crafted shader [Bou+06; Bou+07; Bén+10; Mon+18] to mimic given artistic medium; others compose the result from a library of predefined pen [Sal+97; Pra+01; Sna+06], hatch [Bre+07], or brush strokes [Lit97; HE04; Sch+11; ZZ11]. Although these techniques can deliver convincing results, they work only on their respective domain; they are limited to a single style or a certain artistic tool.

Sloan et al. [Slo+01] tried to address this lack of control over the appearance in their technique called The Lit Sphere (a.k.a. MatCap). They allow the user to prepare a hand-drawn exemplar that depicts a stylized counterpart of an illuminated sphere and use it to stylize the illumination of an arbitrary target 3D model. To do that, they employ environment mapping [BN76]—a particular variant of texture mapping where vertex normals are used for texture lookup instead of UV coordinates. Nevertheless, MatCap cannot be directly applied in our scenario since it assumes the stylization of illumination. Debevec et al. [DTM96] proposed a similar technique that can re-project photographs on 3D models. Although their method is directly applicable in our scenario, it cannot handle more extensive viewpoint changes and it distorts the planar structures in the original style exemplar due to texture re-projection (c.f. Fig. 8.2).

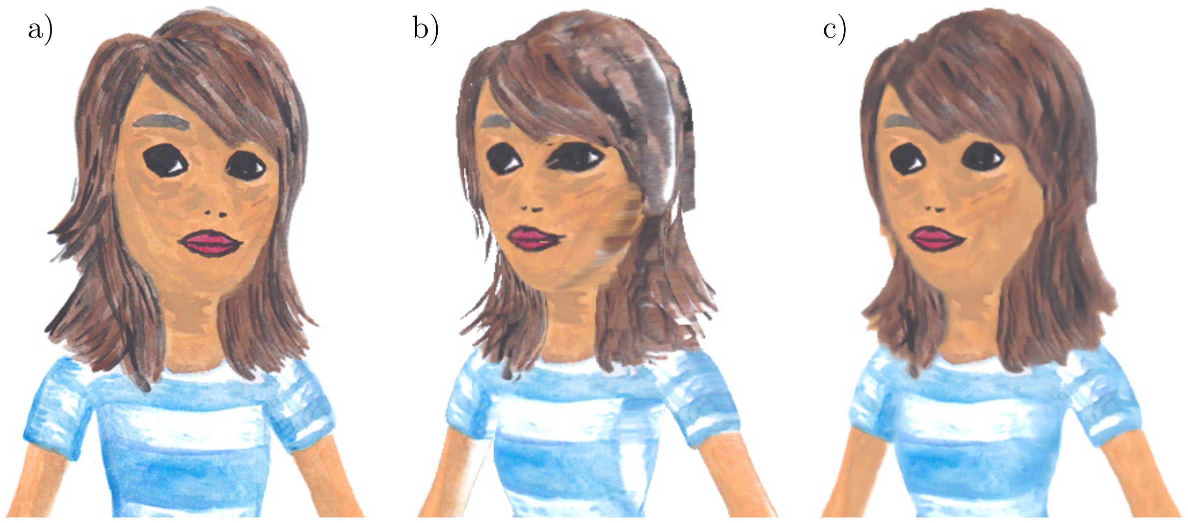


Figure 8.2: *A simple approach to our problem would be to employ the technique of Debevec et al. [DTM96], i.e., to use the original style exemplar (a) as a texture and re-project it on the new pose (b). While this method enables real-time rendering and can provide sufficiently good results when the camera position does not change considerably, it is prone to disturbing artifacts in our scenario. A key issue here is that texture mapping does not preserve the original style exemplar's planarity, i.e., it deforms strokes to respect the shape of the underlying geometry and thus makes the visual system believe the painting was created on the surface and not in the image plane. It is also apparent that the re-projection cannot correctly handle model parts with a normal almost parallel to the original image plane. Since the re-projection is limited to individual triangles, the resulting image may suffer from a misalignment of sharp geometric details with fluffy structures painted in the original style exemplar. Our approach alleviates all mentioned issues (c). Style exemplar (a) courtesy of © Štěpánka Sýkorová.*

Hertzmann et al. [Her+01] proposed an image analogies framework to alleviate the mentioned drawbacks. In their technique, they employ patch-based synthesis [WSI07; Kas+15; Fi16] to preserve the planarity of structures in the original style exemplar while still maintaining meaningful style transfer using additional guiding channels. Others extended this concept to handle style transfer to fluid animations [Jam+15], 3D renders [Fi16], or facial animations [Fi17]. However, obtaining high-resolution stylized images using patch-based synthesis is a computationally expensive task even on the GPU; thus, these methods are hardly accessible when a low computational budget is available, e.g., on a mobile device. Recently, Sýkora et al. [Sýk+19] introduced a real-time variant of guided patch-based synthesis that is, however, limited only to a specific type of guidance containing sufficient spatial variation such as surface normal or texture coordinates.

Our setting bears a resemblance to a stylization scenario where the aim is to propagate the appearance of a single stylized keyframe to the remaining animation frames or a video sequence. This approach was pioneered by Bénard et al. [Bén+13], who extended the patch-based method of Hertzmann et al. [Her+01] by a set of auxiliary guiding channels provided by a 3D renderer and by a new optimization scheme that enables the generation of temporally coherent sequences. Recently, Jamriška et al. [Jam+19] proposed a video stylization framework where necessary guiding channels are extracted automatically from the video. Moreover, Jamriška et al. offer a post-processing step to merge content stylized from different keyframes. However, a fundamental limitation of these techniques is that they are not interactive and can preserve coherency only in one dimension—in time.

A popular example-based approach to style transfer pioneered by Gatys et al. [GEB16] uses the response of the VGG-19 network [SZ14] to measure the similarity of the stylized image and the target content. Based on this measurement, they refine the output stylized image using back-propagation. This approach, however, requires costly optimization. Others used this technique to generate a larger dataset and train a feed-forward network that can reproduce a particular artistic style notably faster [JAFF16; UVL16; Uly+16b; Wan+17; UVL17; WRB17]. However, those approaches suffer from two significant drawbacks: (1) they often fail in reproducing fine textural details presented in the original style exemplar, and (2) they do not guarantee that the transfer is semantically meaningful, e.g., that the strokes used to stylize an eye in the original style exemplar are used to stylize an eye region in the target image.

One can solve the problem of appearance transfer by employing generative adversarial networks [Goo+14]. Those can be trained to perform so-called image-to-image [Iso+17; Zhu+17a; Zhu+17b] as well as video-to-video [Tul+18; Wan+18b] translation. However, this approach relies on a huge dataset of translation pairs, which is not available in our scenario. Some techniques utilize an encoder-decoder scheme to enable the transfer of an arbitrary style to a content image using a single network trained on unpaired exemplars [HB17; Li+17; Lu+17]. The encoder, usually a set of convolutional layers of the VGG-19, extracts feature representation from both style and content image. The features are then combined, and a pre-trained decoder turns them back into the image space. Recently, Kotovenko et al. [Kot+19b; Kot+19a] proposed complex encoder-decoder systems that can deliver impressive results nicely reproducing even lower-level details. Nevertheless, their transfer is still not semantically meaningful as they measure only statistical correlations between the stylized image and the original style exemplar.

Various methods combine aspects of patch-based synthesis and neural-style transfer to achieve semantically meaningful transfer while maintaining neural networks' ability to generalize. To better reproduce local features, Li et al. [LW16b] search for neural patches in a style image while following the structure of a content image. Liao et al. [Lia+17] extended this idea into a deep image analogy framework. Instead of image patches, they compute dense correspondences in the feature space of responses given by the VGG-19 network. Although this technique delivers impressive results, it is computationally expensive and does not support coherence when considering animation. Futschik et al. [Fut+19] approximate the patch-based method of Fišer et al. [Fi17] by training a feed-forward network on a large dataset produced by the mentioned method. Recently, Texler et al. [Tex+20a] combined neural style transfer with patch-based synthesis to enable the generation of high-resolution stylized imagery. Although their approach can deliver notably better stylization quality, it still relies on the network's capability to provide meaningful results respecting scene semantics.

When performing style transfer to animation or video, the temporal consistency has to be taken into account. Although a certain amount of temporal flicker is natural for traditional hand-colored animations [Fi14], it can be visually demanding when the resulting sequence is observed for a longer period and can cause dizziness we would like to avoid. Various methods, both patch-based [Bén+13; Fi17; Dvo+18; Jam+19; Fri+19] and neural-based [Che+17; Gup+17; San+18; RDB18], allow for enforcing temporal consistency explicitly by considering relations between individual animation/video frames. Alternatively, one can employ a blind temporal coherency [Lai+18] to stabilize the arbitrary input video sequence. Although all mentioned methods can help to suppress or entirely remove temporal flicker, they consider temporal coherency only in one dimension—in time. In our scenario, we need to solve the problem of temporal consistency in two or more dimensions.

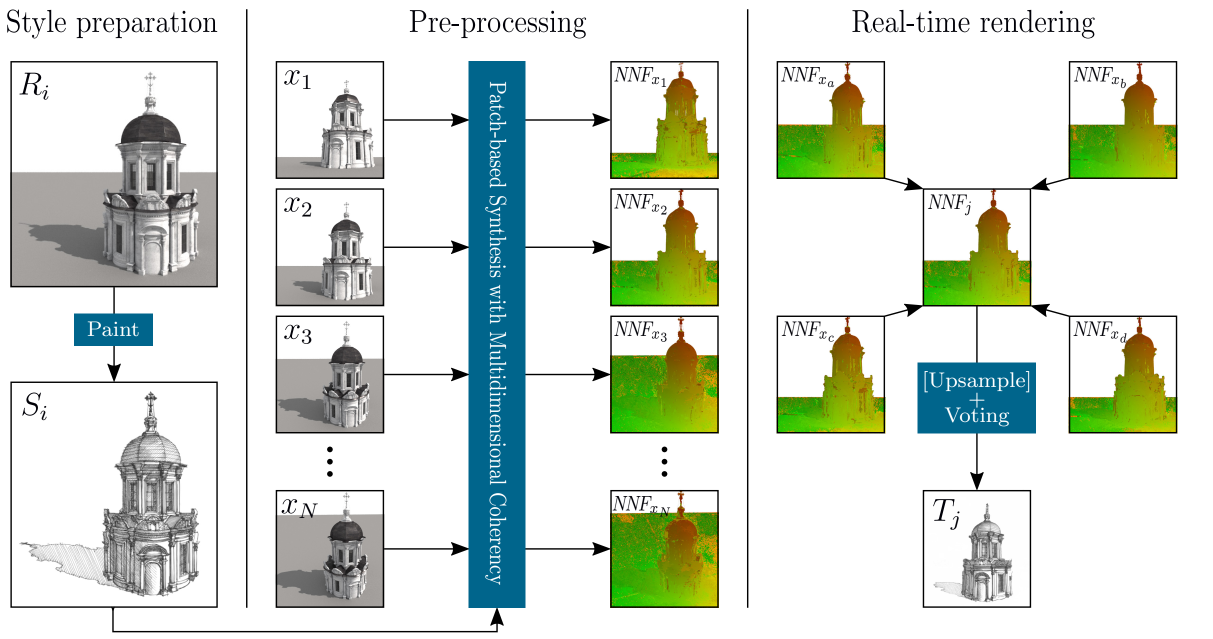


Figure 8.3: *An overview of our method: First, a model M is rendered in preselected interaction state i to produce a stencil R_i over which an artist paints the style exemplar S_i. Also, a set of source guiding channels G_S is rendered at the state i. Then in the pre-processing phase, available interaction space $\mathcal{I}$ is sampled to a set of states $\mathcal{X} = \{x_1, \ldots, x_N\} \subset \mathcal{I}$ and for each such state, the full render, as well as other guiding channels in G_T, are computed. Those serve as an input to our patch-based synthesis algorithm that maintains coherence in multiple dimensions, i.e.; it takes into account consistency between nearby interaction states in X. This algorithm's output is a set of nearest neighbor fields (NNF) at each interaction state x. Those provide a latent representation from which the corresponding stylized image T can be reconstructed (not shown in this figure). Finally, in the real-time rendering phase, the user browses to an arbitrary interaction state $j \in \mathcal{I}$, and at that location, pre-computed NNFs of nearby states $x_a, \ldots, x_d$ are combined to produce NNF_j from which the final target image T_j is reconstructed using voting operation. Alternatively, the NNF upsampling technique of Texler et al. [Tex+20a] can be used to increase the resolution of the output image. See the text for further details. Style exemplar S_i courtesy of © Jan Pokorný.*

Our approach also resembles image-based rendering that can produce impressive novel views from a sparse set of input photographs [Sri+19; Mil+19]. A key difference in our scenario is that we have only a single input image and we aim to preserve planar structures of the original style exemplar, i.e., to retain scale and orientation of individual brush strokes and specific canvas patterns or paper grain. Those features are usually distorted by out-of-plane deformations, which are desirable when generating novel views under perspective projection. These deformations are, however, unwanted in our style transfer scenario.

8.3 Our Approach

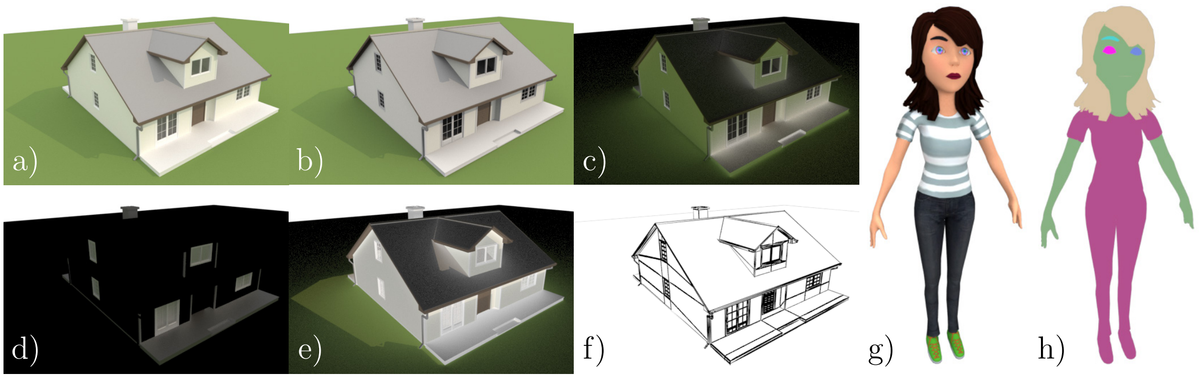


Figure 8.4: *For buildings we used the following set of guiding channels: full global illumination (a), direct (b), indirect (c), specular (d) components, shadow guide (e), and edge guide (f). For characters, to distinguish between different body parts, we used material ID (h) together with full global illumination (g).*

Our method's input is a 3D model M with a texture T that highlights semantically essential details. To prepare a style exemplar, we first produce a render of M: R_i (see Fig. 8.3) at a specific location $i \in \mathcal{I}$, where $\mathcal{I}$ is an interaction space through which the user can explore the model M (e.g., a set of all possible camera rotations or zooming in/out). We assume R_i contains all important structures that would appear when exploring $\mathcal{I}$. In our current implementation i is chosen manually by the user. However, we envision an automatic estimation of the optimal location as future work. Finally, we print R_i on a paper and provide it to the artist as a stencil to prepare a stylized hand-drawn exemplar S_i (also denoted as S). Optionally, the artist can paint over the stencil digitally using a tablet.

The task for our method is to render T_j (see Fig. 8.3)—a stylized counterpart of the target model M seen from a different location $j \in \mathcal{I}$. We would like T_j to be still perceived as a painting/drawing on a canvas/paper comparable to S_i, i.e., we need to preserve planar structures typical for the used artistic media such as brush strokes or canvas pattern. In addition, we would like to retain the artist's intention, i.e., stylize a particular feature in T_j in a similar way as it was stylized in the original style exemplar S_i. Finally, our aim is to render T_j in real-time on a mobile device while maintaining temporal consistency during the interactions in the available interaction space $\mathcal{I}$.

We approach this task in two steps. First, we generate N samples of the interaction space $\mathcal{I}$, i.e., $\mathcal{X} = \{x_1, \ldots, x_N\} \subset \mathcal{I}$. Then for each sample $x \in \mathcal{X}$, we synthesize T_x

using a patch-based synthesis algorithm that respects planar structures of the original style exemplar S_i. Moreover, we also ensure that when a user moves from a sample x_k to a nearby sample x_l, the transition will be visually consistent. Finally, we store a latent representation of T_x denoted as NNF_x, and during the interactive exploration when the user browses through $\mathcal{I}$ into a location j, we retrieve NNFs of all nearby samples around j: $\mathcal{N}_j \subset \mathcal{X}$ and use them to quickly reconstruct the stylized image T_j. A key advantage of combining latent representations instead of blending images is that the final stylized image will look comparable to the original patch-based synthesis algorithm's output.

The task described above has a substantial difference compared to previous patch-based synthesis techniques [Bén+13; Fi17; Jam+19] where the coherence is maintained only in one dimension—in time. In our scenario, we need to achieve consistency in all possible dimensions of $\mathcal{I}$. To do that, we extend the patch-based synthesis algorithm of Fišer et al. [Fi16] (StyLit) to support multidimensional coherence. We provide a brief overview of the original StyLit algorithm, and then we propose its extension.

8.3.1 StyLit algorithm overview

In its original form, StyLit algorithm aims to minimize the following error over all patches in the target synthesized image T:

$$\mathcal{E}(S,T,G_S,G_T) = \sum_{q \in Q_T} \min_{p \in Q_S} \left(E_t(S,T,p,q) + E_g(G_S,G_T,p,q) \right). \tag{8.1}$$

Here Q_S & Q_T are sets of patches in the source style exemplar S and the target synthesized image T, E_t is the texture coherence error:

$$E_t(S,T,p,q) = \|S(p) - T(q)\|^2 \tag{8.2}$$

and E_g is the guidance error:

$$E_g(G_S,G_T,p,q) = \|G_S(p) - G_T(q)\|^2 \tag{8.3}$$

where G_S & G_T are source and target multichannel guides, which are computed using a 3D renderer. Besides a full global illumination channel, its direct/indirect/specular components, and shadow channel (used in the original StyLit algorithm), we added an edge channel (c.f. Fig. 8.4a–f):

$$G_{\{S,T\}} = \{\mathit{full}, \mathit{direct}, \mathit{indirect}, \mathit{specular}, \mathit{shadow}, \mathit{edge}\}. \tag{8.4}$$

For 3D characters we use the full global illumination channel and the material ID channel (c.f. Fig. 8.4g–h):

$$G'_{\{S,T\}} = \{\mathit{full}, \mathit{id}\}. \tag{8.5}$$

To compute $\mathcal{E}$, the nearest neighbor field (NNF) is constructed between the sets of source and target patches Q_S & Q_T. NNF is a look-up table in which each target patch $q \in Q_T$ has stored coordinates of its corresponding source patch $p \in Q_S$. The p corresponds to q if it has the lowest sum of style and guide errors E_t & E_g among all patches in Q_S. Also, during the retrieval of p, an allowable error budget is taken into account to prevent some source patches from being assigned too often as the closest ones (please refer to Fišer et al. [Fi16] for detailed description).

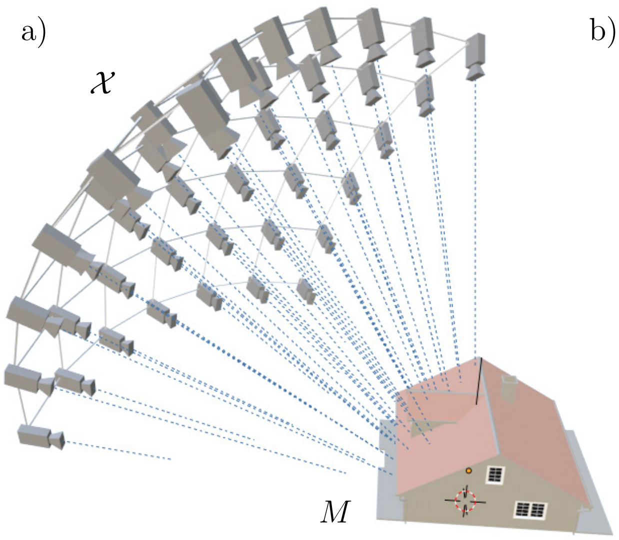
To obtain the final stylized image T, the StyLit algorithm uses an iterative EM-like algorithm initially proposed by Wexler et al. [WSI07]. It alternates two steps: First, in *search step*, NNF is constructed between the source patches Q_S and target patches Q_T. Then in *voting step*, an updated version of T is reconstructed using NNF by computing a weighted average of all co-located pixels from corresponding source patches.

To maintain temporal coherence in Fišer et al. [Fi17] and later in Jamriška et al. [Jam+19], the error (8.1) was extended by an additional temporal coherence term:

$$E_c(S, T', p, q) = \|S(p) - T'(q)\|^2 \tag{8.6}$$

where T' is a previously synthesized frame that was shifted to match with the position of the current frame T. Therefore, the extended error $\mathcal{E}$ which is minimized looks as follows:

$$\begin{aligned}&\mathcal{E}(S, T, T', G_S, G_T) = \\ &\quad \sum_{q \in Q_T} \min_{p \in Q_S} \big(E_t(S, T, p, q) + E_g(G_S, G_T, p, q) + E_c(S, T', p, q)\big).\end{aligned} \tag{8.7}$$



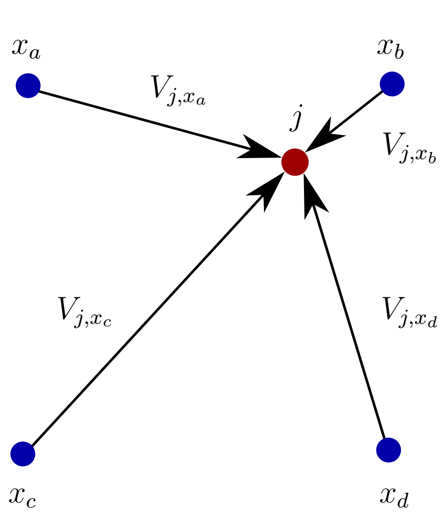


Figure 8.5: *An illustration of a discrete subset $\mathcal{X}$ of an interaction space $\mathcal{I}$ where the pre-calculation of stylized images T_x of the target model M is performed (a). In order to reconstruct a target image T_j in arbitrary location j within the interaction space $\mathcal{I}$, a latent representation NNF_x of nearby images T_x at locations $\{x_a, \ldots, x_d\}$ are shifted towards j using motion vectors $\{V_{j,x_a}, \ldots, V_{j,x_d}\}$ and combined to produce NNF_j from which the target image T_j is subsequently reconstructed (b). See the text for detailed description.*

8.3.2 Multidimensional coherence

The error $\mathcal{E}$ requires the motion-compensated version of the previous frame T' to perform the evaluation. However, in our scenario, we do not have a sequence of frames, but a multidimensional space of all possible interaction states $\mathcal{I}$ (see Fig. 8.5a). To address

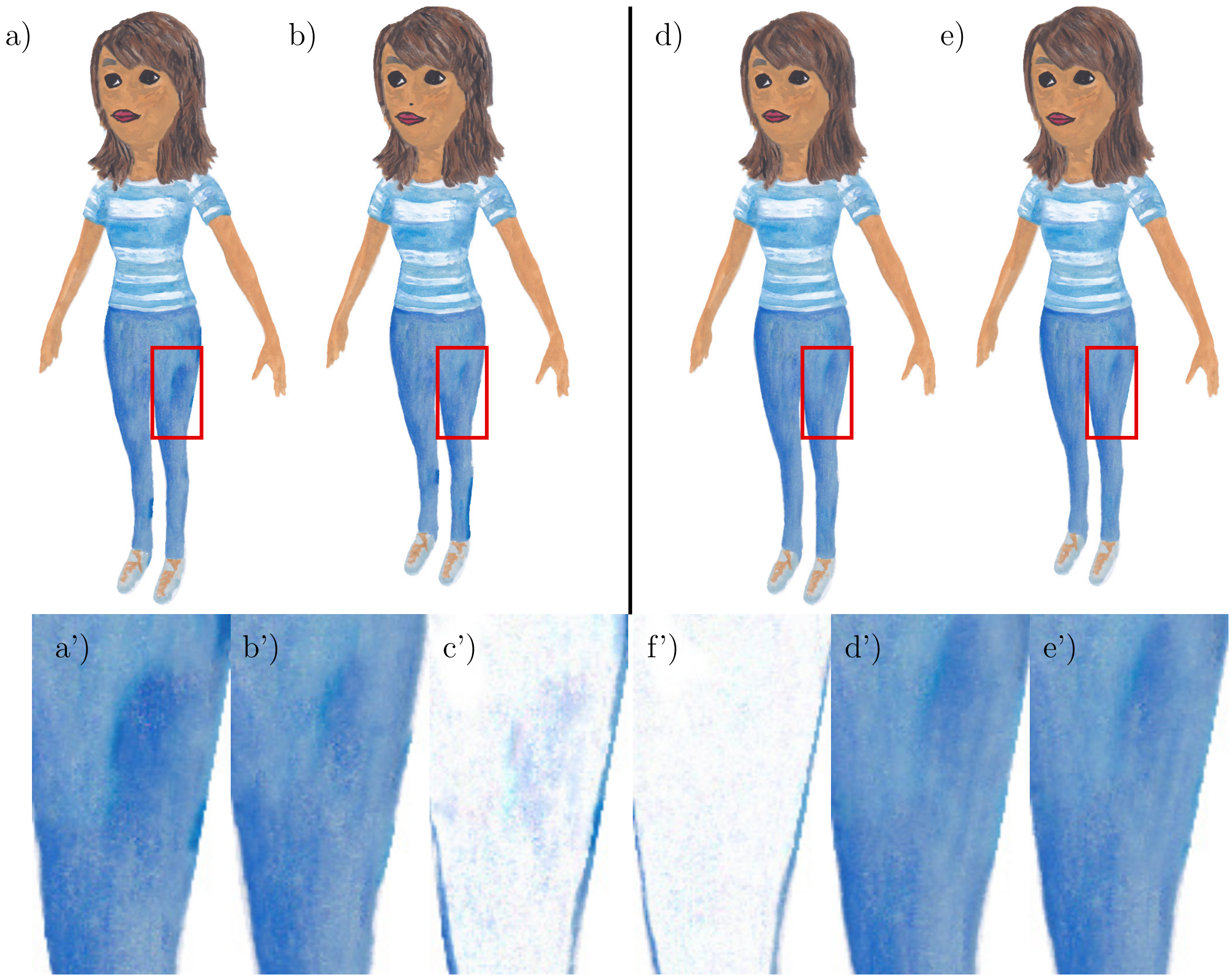


Figure 8.6: *Demonstration of coherence enforcement: (a, b) two consecutive states stylized without handling the coherence, (d, e) the same states with coherence enforced. The area in a red rectangle is enlarged below; notice the difference in (a', b'), while (d', e') appear identical; (c', f) visualize an inverted subtraction of (a') from (b') and (d') from (e'), respectively. Note that (f') is almost white (almost zero difference). Although these differences might not seem prominent on still images, they can be distracting in motion (c.f. our supplementary video).*

such a multidimensional coherence problem, we sample $\mathcal{I}$ to $\mathcal{X} = \{x_1, \ldots, x_N\} \subset \mathcal{I}$ and start the computation of all T_x in parallel. During each search-vote iteration, we get an intermediate stylized result of T_x and warp it to all neighboring interaction states $\mathcal{N}_x \subset \mathcal{X}$. To do that, we leverage the existence of the underlying 3D model to generate accurate motion fields $V_{x,n}$ that capture movement of individual pixels between nearby interaction states. Such a shifted result T'_n is then used as a new coherence guide, i.e., our goal is to minimize a joint error computed over all sampled interaction states $\mathcal{X}$:

$$\sum_{x \in \mathcal{X}} \sum_{n \in \mathcal{N}_x} \mathcal{E}(S, T_x, T'_n, G_S, G_T). \tag{8.8}$$

The importance of this coherence enforcement is demonstrated in Fig. 8.6a–b where two consecutive stylized frames are synthesized without coherence enforcement while in Fig. 8.6d–e coherence is enforced. In the zoom-in patches, significant changes between Fig. 8.6a' and Fig. 8.6b' are visible, but Fig. 8.6d' and Fig. 8.6e' appear almost

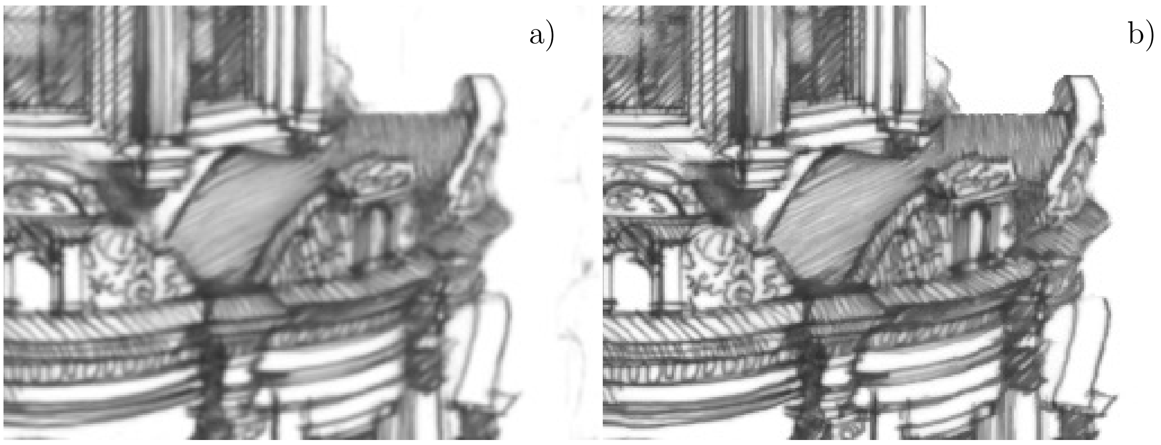


Figure 8.7: *Improving spatial coherency of a combined NNF_j^*: (a) an image T^* produced from NNF_j^* that was combined from nearby pre-computed states of interaction space $\mathcal{N}_j^*$, note blurriness caused by spatial incoherency of NNF_j^*, (b) a sharper image T produced from a refined NNF_j that has better spatial coherency (see the text for a detailed description).*

identical. Even though the changes might not seem very prominent on still images, they could be distracting while in movement (c.f. our supplementary video).

8.3.3 Real-time rendering

The algorithm described in the previous section outputs a coherently stylized model for each sample x of sparsely sampled interaction space $\mathcal{X}$. However, for our target interactive application we need to reconstruct a stylized image T_j at an arbitrary location $j \in \mathcal{I}$. Moreover, we need to retain the visual quality of the original synthesis algorithm, i.e., T_j needs to be a mosaic of larger bitmap chunks taken from the original style exemplar S_i. Therefore, instead of doing some kind of blending operation on the stylized images T_x, we leverage the existence of NNF_x that were used to generate T_x.

We again use the underlying 3D geometry to generate motion vectors $V_{j,x}$ that we use to shift nearby pre-computed NNF_x to a position of the current interaction state $j \in \mathcal{I}$ (see Fig. 8.5b). To generate the combined NNF_j^* at every target pixel t, we set $NNF_j^*(t) = NNF_{\hat{x}}(t - V_{j,k}(t))$ where $\hat{x}$ is the most suitable sample from the set of nearby states $\mathcal{N}_j$. To select $\hat{x}$, we first exclude states that have different object IDs, i.e., that lie outside the object located at the pixel t. Those will form a set of feasible samples $\mathcal{N}_j^*$. Then we generate a small random displacement vector r that is unique for each target pixel t and does not change during the interaction. Finally, we pick $\hat{x}$ such that:

$$\hat{x} = \arg\min_{x \in \mathcal{N}_j^*} \|x - j + r\| \tag{8.9}$$

Such a perturbated closest sample selection does not introduce additional flicker and helps to avoid larger abrupt swaps when the sample x changes.

Since the combined NNF_j^* mixes pixel coordinates from different NNFs of nearby interaction samples it may suffer from lower spatial coherency, i.e., contain smaller coherent chunks when compared to the original NNFs. This may lead to blurring artifacts when NNF_j^* is applied directly in the subsequent voting step (see Fig. 8.7). To improve spatial coherency of the resulting NNF_j, we first apply voting step to obtain an intermediate blurred version of T_j denoted as T_j^*. Then for each patch $q \in Q_{T^*}$ at a pixel t we compute the error $E_c(S, T^*, p, q)$ for every $p \in Q_S$ of with coordinates given by the shifted $NNF_x(t - V_{j,x}(t))$. The coordinates of a patch p with the lowest error E_c are then stored to the refined NNF_j that is used for final voting step to produce T_j. The improvement caused by this refinement can be seen in Fig. 8.7.

8.4 Results

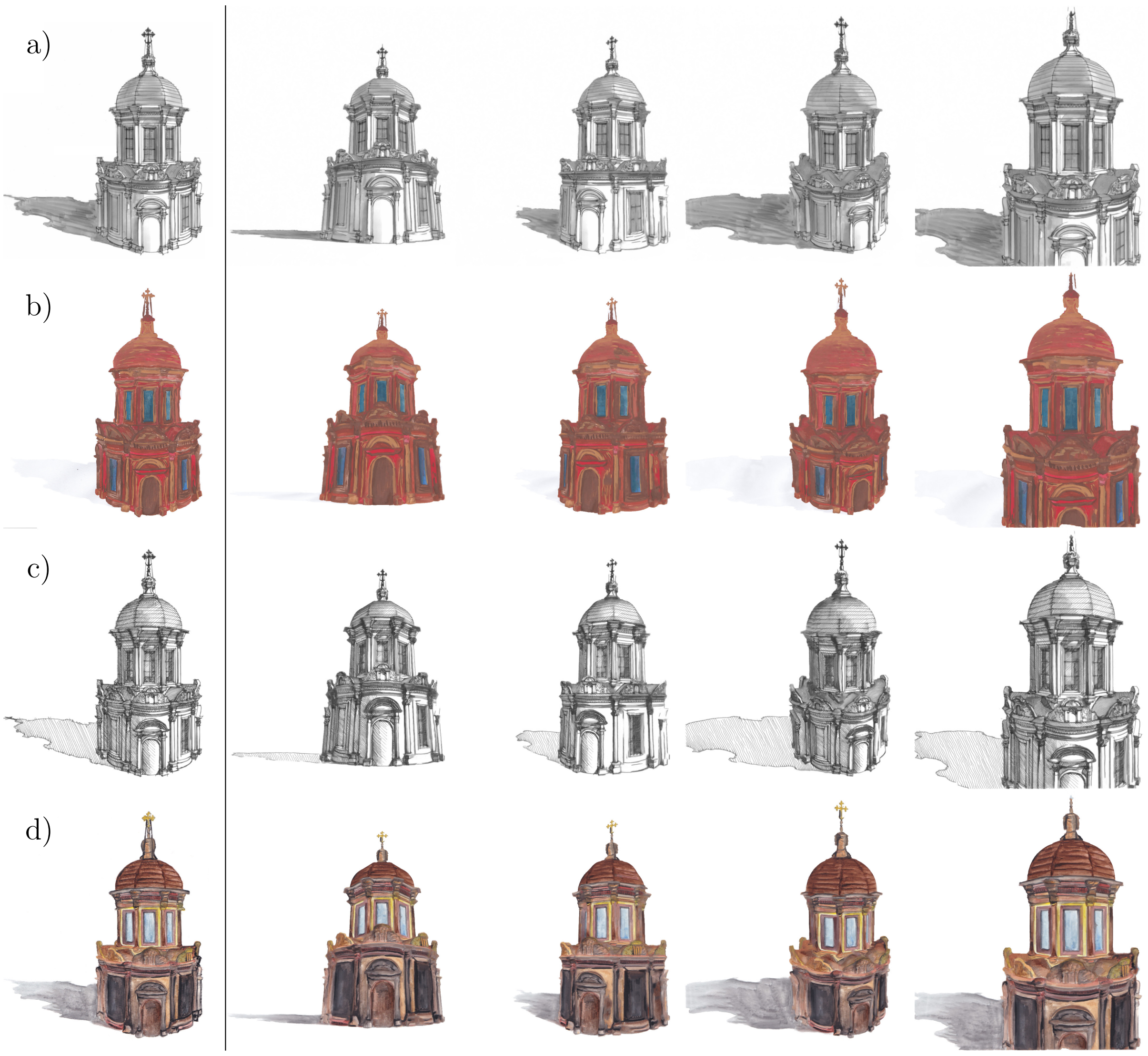


Figure 8.8: *A complex architectonic model of a chapel stylized using markers (a, c) and watercolor (b, d) style exemplars (left) from various viewpoints using our method. The camera is rotating and zooming in/out (right). Note how important planar structures (such as individual pen/brush strokes or a paper grain) typical for the corresponding artistic media are preserved in each result. For the stylization in motion, please, refer to our supplementary video. Style exemplar* (a, c) *courtesy of* © *Jan Pokorný and* (b, d) © *Štěpánka Sýkorová.*

We implemented our patch-based synthesis algorithm in C++ and CUDA. To generate guiding channels, we use GPU implementation of [Kaj86]. The overall computation

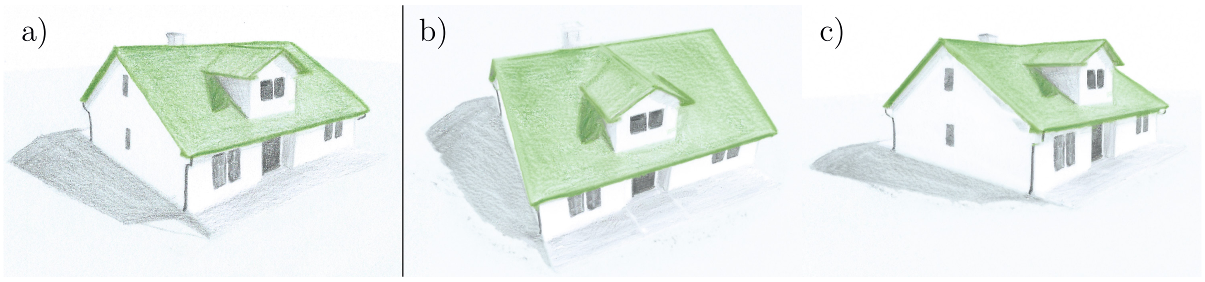


Figure 8.9: *A model of family house painted using color pencils (a). Results of our method (b, c) faithfully represent the original style exemplar and respect the content of an underlying 3D model. The stylized model can be viewed on computer or mobile phone in real-time (see our supplementary video). Style exemplars (a) courtesy of © Barbora Kociánová.*

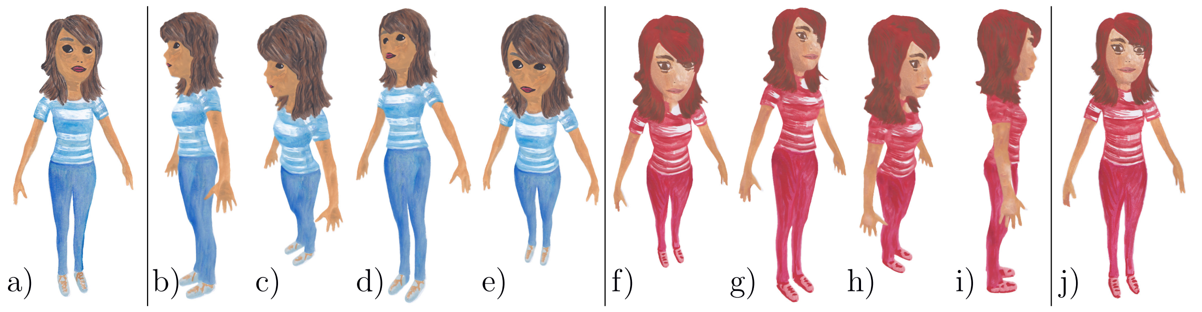


Figure 8.10: *A model of a girl stylized using two different watercolor styles (a, j). The results (b–e) were produced using style (a) and results (f–i) using style (j). Even in extreme poses, stylized images (b) and (i) retain the content of an underlying 3D model well. The stylized model can be viewed on desktop computer as well as on a mobile phone in real-time, see our supplementary video. Style exemplars (a, j) courtesy of © Štěpánka Sýkorová.*

(guiding channels and synthesis) takes around 10 seconds for one interaction sample with resolution of 1280x720 on Nvidia RTX 2080 GPU. For $\mathcal{I}$ where camera is rotating around the model (see Fig. 8.5) in a range of 180 degrees for horizontal direction and 50 degrees for vertical direction with sampling rate 10 degrees we get 90 samples of $\mathcal{X}$ that can be generated in less than 15 minutes. The second part of our approach—NNF merging and rendering is implemented in Unity framework using HLSL shaders that can fully utilize GPU. Thanks to this integration, the renderer can easily be deployed on desktop machines as well as on a wide range of mobile devices (c.f. Fig. 8.1i–j).

Resulting NNFs are stored as 2D arrays where each entry contains two coordinates (short integers). After the LZMA compression (in Unity), such a lossless latent representation is smaller than the final stylized image stored in PNG format or roughly the same size as a medium–high quality JPEG image of the same resolution. The compressed bundle of 90 samples takes around 19MB of space.

Both the computation time for guiding channels and patch-based synthesis and the memory footprint can further be reduced by using NNF *upscaling* method of Texler et al. [Tex+20a]. When NNF is upscaled two times, the resulting quality is still acceptable while the computational overhead is reduced to roughly 4 minutes and the size of 90 NNF samples is only 5MB. Thus, the entire interaction space for a new style exemplar can be sampled, stylized, transferred, and viewed on a mobile device relatively quickly.

To evaluate our method, we choose two characters and two architectonic models for which we let artists to prepare different style exemplars using watercolor, markers, color pencils, and chalk. We sampled two different interaction spaces: (1) camera moving around the object and (2) camera moving in a horizontal direction and zooming in/out. In Fig. 8.8 we show results on a architectonic model of chapel stylized in four different artistic styles. Another architectonic model is shown in Fig. 8.9. In Figures 8.1, 8.10, and 8.11 we present results on two different character models. To demonstrate the potential of our approach to be executed in real-time on a mobile device, in Fig. 8.1i–j we show our method running on Samsung Galaxy Note 8 at 20 frames per second. For full recordings of real-time interaction sessions please refer to our supplementary video.

An important parameter of our method is the sampling rate of the available interaction space (e.g., angular difference between nearby camera viewpoints). In Fig. 8.12 we compare results created using four different sampling rates, their memory requirements, and the time of pre-calculation. The visual quality difference between the sampling rate of 2 and 5 degrees is almost negligible. For sampling rate of 10 degrees some blurring artifacts start to show up, however, those are not visible on small screens, e.g., mobile phones, and thus 10 degrees can serve as a good compromise between visual quality, storage space and computational overhead. The results with sampling rate of 20 degrees and more already show considerable artifacts.

In Fig. 8.13 and in our supplementary video we compared our approach with the current state-of-the-art in patch-based synthesis as well as with some neural-based techniques.

The seminal example-based method of Bénard et al. [Bén+13] (Fig. 8.13b) as well as the current improvement of Jamriška et al. [Jam+19] (Fig. 8.13c) were designed for image sequences, therefore, they suffer from discontinuities when browsing in multidimensional interaction space. We used those previous techniques to illustrate this limitation and precalculate a few linear trajectories over the entire interaction space. During the viewing session, we then let the user navigate freely in the interaction space, pick the closest pre-

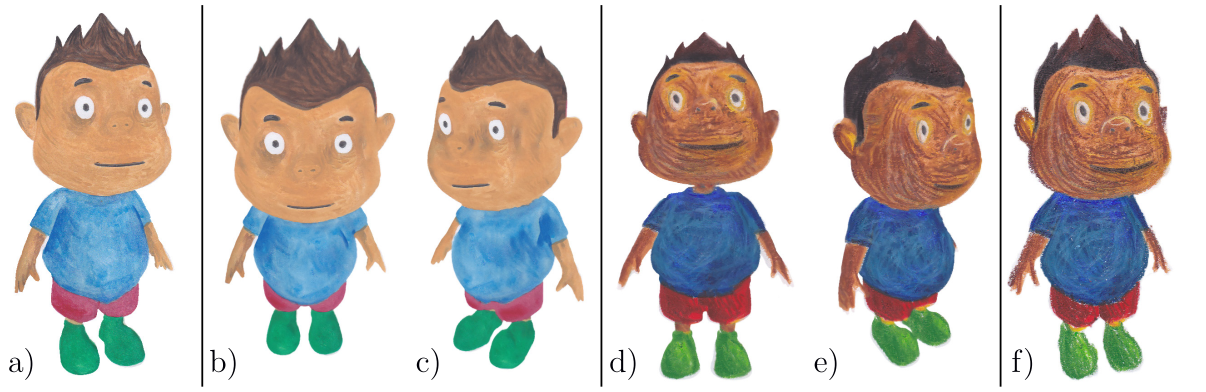


Figure 8.11: *A model of a boy stylized using watercolor (left) and chalk (right). Our results (b–e) faithfully mimic the original style exemplars (a, f), preserving the notion of a painting/drawing created by hand on the paper (c.f. our supplementary video for the model in motion). Style exemplars (a, f) courtesy of © Štěpánka Sýkorová.*

computed path, and replay its frames as long as its direction remains similar to the user's intent. In the case when the user starts navigate differently, we pick another trajectory that is closer to a new path. Due to the one-dimensional coherence, such a hard jump leads to abrupt changes in the appearance, as is visible in our supplementary video. Performance-wise the method of Bénard et al. took several minutes per frame to compute and thus is not applicable in our interactive scenario. The method of Jamriška et al. [Jam+19] running on the GPU is notably faster (few seconds per frame), however, still not fast enough for interactive use.

The method of Debevec et al. [DTM96] (Fig. 8.13e) and recent approach of Sýkora et al. [Sýk+19] (Fig. 8.13d) can run at interactive rates, however, since they use texture mapping coordinates to perform the re-projection / style transfer, they fail to handle parts of the model which are not properly stylized in the original exemplar S_i. Although the StyleBlit algorithm can use additional guides (such as object IDs) that are less restrictive and allow for better generalization, one local guide still needs to remain in the set of guiding channels to satisfy the StyleBlit requirements. To perform a meaningful comparison using our guiding channels (Fig. 8.4g–h), which are not local, we had to add a local guide, i.e., texture mapping coordinates. Due to this reason, the results shown in Fig. 8.13d suffer from similar artifacts as the method of Debevec et al. [DTM96]. An essential advantage of our approach is that it could potentially work with any guiding channels as the original StyLit algorithm [Fi16].

Neural-based techniques are slow to compute (tens of seconds per frame) and in general have difficulties to preserve important high-frequency details of the original artistic media as is visible in the output of Li et al. [Li+17] (Fig. 8.13f) and Gu et al. [Gu+18] (Fig. 8.13h). While deep image analogies [Lia+17] (Fig. 8.13g) performs better with respect to high-frequency details, they cannot properly handle temporal coherence.

8.5 Limitations and Future Work

Although our approach enables interactive exploration of a stylized 3D model on a mobile device while faithfully reproducing unique visual characteristics of the used artistic media and preserving temporal coherency, there are still some limitations that could motivate future work.

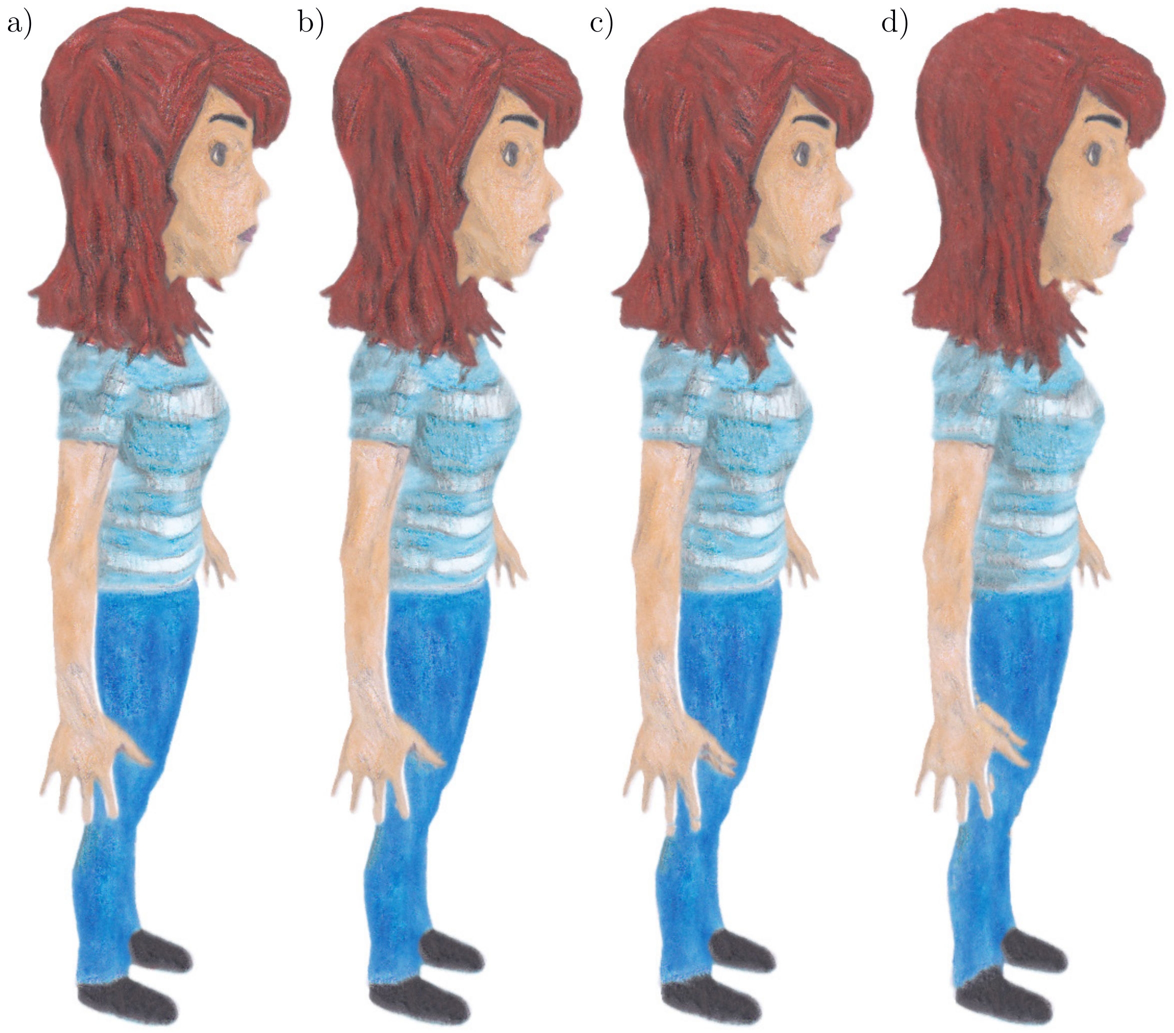
As our technique uses guided patch-based synthesis [Fi16] it also shares its drawbacks. The style exemplar needs to be aligned relatively well with the original render, i.e., notable discrepancies (e.g., shape caricature) may lead to a structural mismatch. Tiny details such as nostrils may occasionally disappear due to relatively small spatial support. For those parts, adding a specific guide would be beneficial. Although the original algorithm [Fi16] handles brush strokes crossing the object boundaries, in our real-time NNF combination phase, the object ID masking mechanism may lead to visible discontinuities. Special handling would be necessary to preserve the appearance of structured boundaries.

Despite the fact that our approach explicitly handles multidimensional coherence, it may not always achieve fully coherent results. Since the result of patch-based synthesis is a seamless mosaic of small, translated chunks of the original style exemplar, occasional popping is inevitable. This effect was also apparent in previous patch-based methods (see, e.g., [Jam+19]) where it can bring the notion of hand-colored sequence [Fi14] but it may also introduce unwanted distraction. In future work, we plan to control it by combining patch-based and neural techniques.

Our method can suffer from significant artifacts when executed on a larger interaction space where, e.g., the camera viewpoint differs significantly from the one used for creation of style exemplar S. The synthesis then fails to find appropriate exemplar patches for the unseen content and it starts to use patches from inappropriate areas. Since the coherence is enforced in all dimensions, the synthesis may propagate those errors across the entire interaction space. Due to this reason artifacts from distant interaction states (w.r.t. S) diffuse to nearby states which would otherwise be stylized properly if a smaller interaction space is used. This limitation is illustrated in Fig. 8.14 (and in our supplementary video) where two samples from the same viewpoint are displayed side-by-side: one is generated from a larger interaction space that goes beyond the limits of our method and the other uses a smaller space.

8.6 Conclusion

We introduced an interactive approach to the stylization of 3D models that faithfully reproduces a given hand-drawn exemplar while preserving coherence during its exploration. To allow this, (1) we designed a novel variant of a patch-based synthesis algorithm that can produce a sparse set of samples from the available interaction space. Those are produced in a way that all nearby states are stylized coherently. Then, during the real-time rendering phase (2), we demonstrate how to swiftly combine those pre-calculated samples to produce the final stylized image at an arbitrary location. Thanks to this two-stage approach, a real-time 3D model exploration is feasible even on a mobile device. We verified our method on various 3D models and hand-drawn styles and compared them with the current state-of-the-art.



2 degrees	5 degrees	10 degrees	20 degrees
341 samples	65 samples	21 samples	8 samples
65.04 MB	13.85 MB	5.85 MB	3.46 MB
3491 sec	682 sec	226 sec	88 sec

Figure 8.12: *Comparison of four different sampling rates. From left to right, dense sampling to sparse sampling; (a) sampled every 2 angular degrees, (b) 5 degrees, (c) 10 degrees, and (d) 20 degrees. Sampling rate defines trade-off between quality and performance, i.e., with dense sampling the quality is high, however, time required to run the patch-based synthesis and size of the package might be intractable. Compare the visual quality of (a) and (d) and their respective memory and computational time requirements. We found that sampling rate of 5 or 10 degrees is a good compromise.*

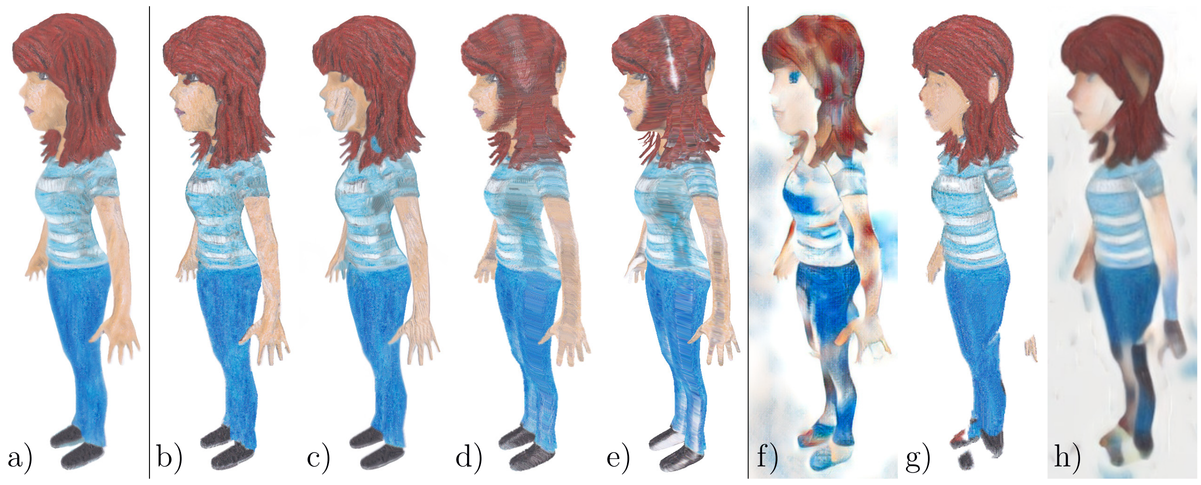


Figure 8.13: *Comparison of our approach (a) with other example-based stylization methods. Method of Bénard et al. [Bén+13] (b) and Jamriška et al. [Jam+19] (c) show artifacts due to their inability to maintain temporal coherence in multiple dimensions; Sýkora et al. [Sýk+19] (d) and Debevec et al. [DTM96] (e) fail to stylize parts of the model which are not well covered by texture in the original style exemplar; Li et al. [Li+17] (f) fail to reproduce appearance of the style exemplar; Liao et al. [Lia+17] (g) preserve texture properties faithfully, however, do not maintain global consistency; Gu et al. [Gu+18] (h) yield poor texture as well as content quality. Please, refer to our supplementary video to see this comparison in motion.*

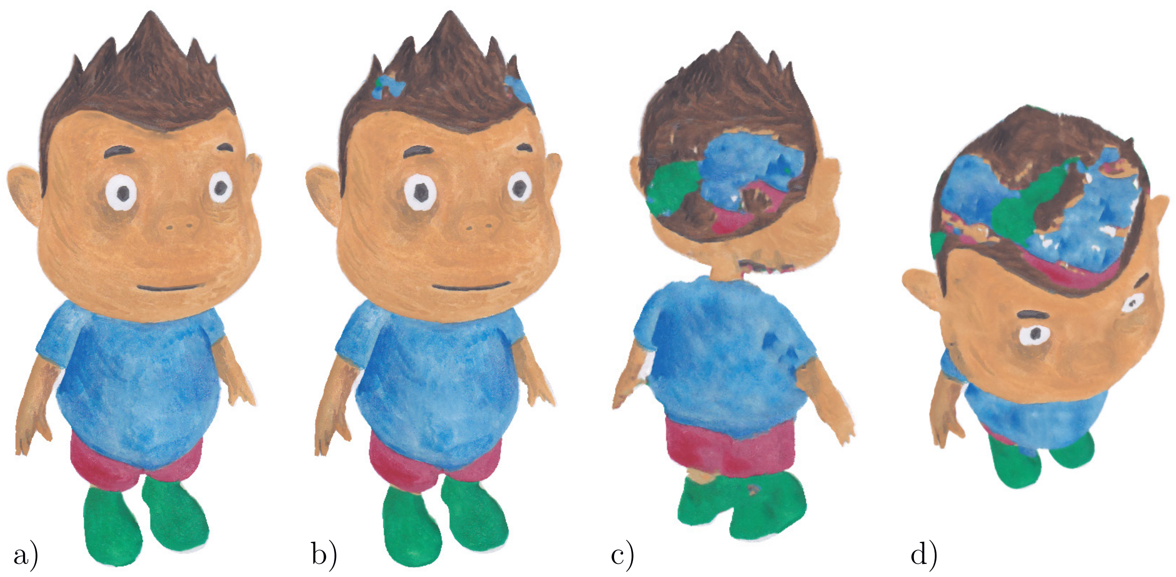


Figure 8.14: *Comparing results computed using large and smaller interaction space: (a) result generated from a small interaction space where all samples are stylized without visible artifacts, (b) result of the same viewpoint as (a), but taken from a synthesis running on a larger interaction space. Note the artifacts on the hair region propagated from samples far from the style exemplar (c, d) that are not stylized correctly due to significant content difference.*

Chapter 9

Conclusion and Future Work

This thesis has presented five novel methods contributing to the example-based style transfer and advancing the current state-of-the-art. Presented techniques allow for scenarios that were hardly possible before, e.g., semantically meaningful style transfer to videos when interactive response is demanded, or learning an image-to-image translation network on the fly to stylize live video stream. Following in this chapter is a summary of the contribution and novelties of our work. Lastly, we mention the concurrent development, and propose topics for future investigation.

9.1 Summary

In Chapter 4 we presented our contribution into the problem of general style transfer [Tex+20a] published in Computers & Graphics journal (previous version published as a conference paper [Tex+19] at Expressive 2019). We combined neural and patch-based style transfer methods. Neural techniques provide us adequate stylization at the global level, and we use their output as a prior for subsequent patch-based synthesis. Thanks to this combination, we were able to keep the high frequencies of the original artistic media better, thereby dramatically increase the fidelity of the resulting stylized imagery. Moreover, we introduced a way to stylize extremely large images, e.g., 340 Mpix, while maintaining high visual quality. Furthermore, we presented a novel stylization algorithm where we directly use responses of object recognition neural network to guide the patch-based synthesis. We showed that this approach is capable of delivering comparable visual quality to state-of-the-art neural style transfer.

In Chapter 5, we introduced our work [Jam+19], that was presented at SIGGRAPH 2019 conference and published in ACM Transactions on Graphics journal. In this work we introduced a new example-based approach to video stylization, with a focus on preserving the visual quality of the style, user controllability and applicability to arbitrary video. To achieve this, we developed a new type of guidance for state-of-art patch-based synthesis, that can be applied to any type of video content and does not require any additional information besides the video itself and a user-specified mask of the region to be stylized. Moreover, in the case multiple stylized keyframes are required, we presented a blending technique to seamlessly transition between them while preserving texture coherence, contrast, and high frequency details.

In Chapter 6, we introduced our publication Interactive Video Stylization Using Few-Shot Patch-Based Training [Tex+20b] presented in SIGGRAPH 2020 conference and published in ACM Transactions on Graphics journal. In this publication, we introduced a new training strategy for image-to-image translation networks, and we successfully used it to the problem of keyframe-based video stylization. We were able to develop a framework that requires only a single training image, the model is trained from scratch in the order of minutes on a consumer-grade GPUs, and inference runs in real-time. Our framework preserves temporal coherency without the need to process previous frames thus allows for random access or parallel processing. Moreover, it also implicitly handles multiple keyframes and it produces consistent results without any explicit blending operation. This allowed us to come up with various interactive scenarios that were not possible before, e.g., a real-time style transfer to a live video stream that uses a captured exemplar painted simultaneously on the canvas.

In Chapter 7, we presented our contribution to stylize 3D renders, StyleBlit: Fast Example-Based Stylization with Local Guidance [Sýk+19], which was presented at Eurographics 2019 conference and published in Computer Graphics Forum journal. We developed an example-based style transfer algorithm capable of delivering high-quality stylization in real-time while demanding minimal hardware requirements. In StyleBlit, we presented a way to bypass expensive optimization steps that are normally required to achieve high-quality style transfer. To achieve this, we employed so-called local guidance that encourages style transfer to be semantically meaningful. This allows us to deploy our approach in scenarios where a low computational budget is available, e.g., mobile phone or games.

In Chapter 8, we presented our latest work, StyleProp: Real-time Example-based Stylization of 3D Models [Hau+20] that was accepted to Pacific Graphics 2020 conference and published in Computer Graphics Forum journal. We introduced a novel approach to the real-time non-photorealistic rendering of 3D models where the appearance is given by a single hand-drawn exemplar. We utilized guided patch-based synthesis to secure high visual quality, and we presented how to maintain temporal consistency in multiple dimensions. We enable interactive experience by precalculating a sparse latent representation of the entire interaction space that is then merged in real-time to create continuous movement. This lightweight approach is capable of running on mobile devices while delivering full stylization experience.

9.2 Concurrent and Future Work

As example based style transfer is popular topic among researchers, many new techniques have been developed along with our research.

Kotovenko et al. [Kot+19a] addressed an important problem in style transfer. Artists usually change their style throughout their career, meaning, some paintings of an artist can be dramatically different from his or her other paintings. Thus it is not practical to train the neural network using a large corpus composed of all artist's paintings. With this in mind, in our research [Jam+19; Tex+20b] we focused on using just one particular painting to perform the style transfer, Kotovenko et al. [Kot+19a] on the other hand, presented a way to capture particularities of style and variations within multiple paintings of the same artist. The same authors contributed to the style transfer also by introducing content transformation module [Kot+19b], that is located between encoder and decoder. The aim is to perform style transfer that reflects not only color or texture, but also deformation or geometrical changes that are part of the style, e.g., some content is added or removed as the style is intentionally omitting or amplifying certain content. Both mentioned methods yield impressive computer-generated artworks, however, the semantically meaningful results are not guaranteed.

Similarly to our efforts to combine traditional computer graphics techniques with neural-based ones [Tex+20a], a method that uses graph cut in a neural-based framework was presented [Zha+19]. They explicitly match the semantic patterns in content and style images. They clustered the style image features into sub-style components, which are then matched with local content features under a graph cut formulation. To render the final result, a reconstruction network is trained to transfer each sub-style to a stylized image. This method works on large variety of styles and is producing state-of-the-art results.

Concurrently with our paper [Tex+20b], a learning based method to transfer the appearance of exemplar SVBRDF to a target image representing similar material was presented [DDB20]. Similarly to our approach, they use a small exemplar-specific dataset to train the network, however, they do not train it from scratch, they start with model pre-trained on generic training set.

Concurrently with our paper [Sýk+19], Friedrich et al. [FM19] used neural style transfer on 3D voxel primitives. Based on the 2D pixel and 3D voxel analogy, they successfully applied style transfer, that is designed to work on 2D RGB images, to voxels—data lacking color, texture, and smooth gradients.

As future work, we envision immense potential in mixing computer vision based style transfer techniques with traditional texture synthesis methods. For instance, bringing other computer vision applications (e.g., segmentation or classification) into the style transfer; this could help to remove necessity of user interaction during the stylization process as well as allow for broader variety of applications. Furthermore, since some style transfer methods are already capable of running at interactive framerate, we consider promising to focus on efficiency, and bring style transfer techniques into the game industry.

In conclusion, style transfer has gotten increased attention in last few years among independent artists as well as animation studios. The methods we presented in this thesis help artists to be more efficient, creative, and enjoy the full potential of modern technologies. The research of advanced algorithms and new hardware paces tremendously fast nowadays; and we believe that style transfer algorithms will soon become an essential part of a toolset that artists and animators use every day.

References

[Ary+98] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. "An optimal algorithm for approximate nearest neighbor searching in fixed dimensions". In: *Journal of the ACM* 45.6 (1998), pp. 891–923.

[Ash01] M. Ashikhmin. "Synthesizing natural textures". In: *Proceedings of Symposium on Interactive 3D graphics*. 2001, pp. 217–226.

[BA83] J. R. Burt and E. H. Adelson. "A Multiresolution Spline With Application to Image Mosaics". In: *ACM Transactions on Graphics* 2.4 (1983), pp. 217–236.

[Bar+09] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. "PatchMatch: A randomized correspondence algorithm for structural image editing". In: *ACM Transactions on Graphics* 28.3 (2009), p. 24.

[Bén+10] P. Bénard, A. Lagae, P. Vangorp, S. Lefebvre, G. Drettakis, and J. Thollot. "A Dynamic Noise Primitive for Coherent Stylization". In: *Computer Graphics Forum* 29.4 (2010), pp. 1497–1506.

[Bén+13] P. Bénard, F. Cole, M. Kass, I. Mordatch, J. Hegarty, M. S. Senn, K. Fleischer, D. Pesare, and K. Breeden. "Stylizing Animation By Example". In: *ACM Transactions on Graphics* 32.4 (2013), p. 119.

[Bha+08] P. Bhat, B. Curless, M. Cohen, and C. L. Zitnick. "Fourier Analysis of the 2D Screened Poisson Equation for Gradient Domain Problems". In: *Proceedings of European Conference on Computer Vision*. 2008, pp. 114–128.

[BHY15] S. Bi, X. Han, and Y. Yu. "An L1 Image Transform for Edge-preserving Smoothing and Scene-level Intrinsic Decomposition". In: *ACM Transactions on Graphics* 34.4 (2015), p. 78.

[BKR17] S. Bi, N. K. Kalantari, and R. Ramamoorthi. "Patch-Based Optimization for Image-Based Texture Mapping". In: *ACM Transactions on Graphics* 36.4 (2017), p. 106.

[BM05] E. P. Bennett and L. McMillan. "Video Enhancement Using Per-Pixel Virtual Exposures". In: *ACM Transactions on Graphics* 24.3 (2005), pp. 845–852.

[BN76] J. F. Blinn and M. E. Newell. "Texture and Reflection in Computer Generated Images". In: *Communications of the ACM* 19.10 (1976), pp. 542–547.

[Bou+06] A. Bousseau, M. Kaplan, J. Thollot, and F. X. Sillion. “Interactive watercolor rendering with temporal coherence and abstraction”. In: *Proceedings of International Symposium on Non-Photorealistic Animation and Rendering*. 2006, pp. 141–149.

[Bou+07] A. Bousseau, F. Neyret, J. Thollot, and D. Salesin. “Video watercolorization using bidirectional texture advection”. In: *ACM Transactions on Graphics* 26.3 (2007), p. 104.

[Bre+07] S. Breslav, K. Szerszen, L. Markosian, P. Barla, and J. Thollot. “Dynamic 2D patterns for shading 3D scenes”. In: *ACM Transactions on Graphics* 26.3 (2007), p. 20.

[Bro+14] M. Browning, C. Barnes, S. Ritter, and A. Finkelstein. “Stylized Keyframe Animation of Fluid Simulations”. In: *Proceedings of International Symposium on Non-Photorealistic Animation and Rendering*. 2014, pp. 63–70.

[BTM06] P. Barla, J. Thollot, and L. Markosian. “X-Toon: An Extended Toon Shader”. In: *Proceedings of International Symposium on Non-Photorealistic Animation and Rendering*. 2006, pp. 127–132.

[BZ17] C. Barnes and F.-L. Zhang. “A survey of the state-of-the-art in patch-based synthesis”. In: *Computational Visual Media* 3.1 (2017), pp. 3–20.

[Cal+19] S. Calvo, A. Serrano, D. Gutierrez, and B. Masia. “Structure-preserving Style Transfer”. In: *Proceedings of Spanish Computer Graphics Conference*. 2019, pp. 25–30.

[Cha+19] C. Chan, S. Ginosar, T. Zhou, and A. A. Efros. “Everybody Dance Now”. In: *Proceedings of IEEE International Conference on Computer Vision*. 2019, pp. 5933–5942.

[Che+13] Z. Chen, H. Jin, Z. Lin, S. Cohen, and Y. Wu. “Large Displacement Optical Flow from Nearest Neighbor Fields”. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 2443–2450.

[Che+17] D. Chen, J. Liao, L. Yuan, N. Yu, and G. Hua. “Coherent Online Video Style Transfer”. In: *Proceedings of IEEE International Conference on Computer Vision*. 2017, pp. 1114–1123.

[Chu+20] M. Chu, Y. Xie, J. Mayer, L. Leal-Taixé, and N. Thuerey. “Learning Temporal Coherence via Self-Supervision for GAN-Based Video Generation”. In: *ACM Transactions on Graphics* 39.4 (2020).

[Cur+97] C. J. Curtis, S. E. Anderson, J. E. Seims, K. W. Fleischer, and D. H. Salesin. “Computer-generated watercolor”. In: *SIGGRAPH Conference Proceedings*. 1997, pp. 421–430.

[Dar+12] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen. “Image Melding: Combining inconsistent images using patch-based synthesis”. In: *ACM Transactions on Graphics* 31.4 (2012), p. 82.

[DDB20] V. Deschaintre, G. Drettakis, and A. Bousseau. “Guided Fine-Tuning for Large-Scale Material Transfer”. In: *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 39.4 (2020).

[Dia+15] O. Diamanti, C. Barnes, S. Paris, E. Shechtman, and O. Sorkine-Hornung. "Synthesis of Complex Image Appearance from Limited Exemplars". In: *ACM Transactions on Graphics* 34.2 (2015), p. 22.

[DSK16] V. Dumoulin, J. Shlens, and M. Kudlur. "A Learned Representation For Artistic Style". In: *CoRR* abs/1610.07629 (2016).

[DTM96] P. E. Debevec, C. J. Taylor, and J. Malik. "Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach". In: *SIGGRAPH Conference Proceedings.* 1996, pp. 11–20.

[Dvo+18] M. Dvorožňák, W. Li, V. G. Kim, and D. Sýkora. "ToonSynth: Example-Based Synthesis of Hand-Colored Cartoon Animations". In: *ACM Transactions on Graphics* 37.4 (2018), p. 167.

[EF01] A. A. Efros and W. T. Freeman. "Image Quilting for Texture Synthesis and Transfer". In: *SIGGRAPH Conference Proceedings.* 2001, pp. 341–346.

[EL99] A. A. Efros and T. K. Leung. "Texture Synthesis by Non-Parametric Sampling". In: *Proceedings of IEEE International Conference on Computer Vision.* 1999, pp. 1033–1038.

[Fi14] J. Fišer, M. Lukáč, O. Jamriška, M. Čadík, Y. Gingold, P. Asente, and D. Sýkora. "Color Me Noisy: Example-based Rendering of Hand-colored Animations with Temporal Noise Control". In: *Computer Graphics Forum* 33.4 (2014), pp. 1–10.

[Fi16] J. Fišer, O. Jamriška, M. Lukáč, E. Shechtman, P. Asente, J. Lu, and D. Sýkora. "StyLit: Illumination-Guided Example-Based Stylization of 3D Renderings". In: *ACM Transactions on Graphics* 35.4 (2016), p. 92.

[Fi17] J. Fišer, O. Jamriška, D. Simons, E. Shechtman, J. Lu, P. Asente, M. Lukáč, and D. Sýkora. "Example-Based Synthesis of Stylized Facial Animations". In: *ACM Transactions on Graphics* 36.4 (2017), p. 155.

[FM19] T. Friedrich and S. Menzel. "Standardization of Gram Matrix for Improved 3D Neural Style Transfer". In: *2019 IEEE Symposium Series on Computational Intelligence (SSCI).* 2019, pp. 1375–1382.

[Fri+16] O. Frigo, N. Sabater, J. Delon, and P. Hellier. "Split and Match: Example-Based Adaptive Patch Sampling for Unsupervised Style Transfer". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition.* 2016, pp. 553–561.

[Fri+19] O. Frigo, N. Sabater, J. Delon, and P. Hellier. "Video Style Transfer by Consistent Adaptive Patch Sampling". In: *The Visual Computer* 35.3 (2019), pp. 429–443.

[Fut+19] D. Futschik, M. Chai, C. Cao, C. Ma, A. Stoliar, S. Korolev, S. Tulyakov, M. Kučera, and D. Sýkora. "Real-Time Patch-Based Stylization of Portraits Using Generative Adversarial Network". In: *Proceedings of the ACM/EG Expressive Symposium.* 2019, pp. 33–42.

[Gat+17] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman. "Controlling Perceptual Factors in Neural Style Transfer". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 3985–3993.

[GEB16] L. A. Gatys, A. S. Ecker, and M. Bethge. "Image Style Transfer Using Convolutional Neural Networks". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2414–2423.

[Goo+14] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. 2014, pp. 2672–2680.

[Gu+18] S. Gu, C. Chen, J. Liao, and L. Yuan. "Arbitrary Style Transfer with Deep Feature Reshuffle". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8222–8231.

[Gui+17] G. Guingo, B. Sauvage, J.-M. Dischler, and M.-P. Cani. "Bi-Layer Textures: A Model for Synthesis and Deformation of Composite Textures". In: *Computer Graphics Forum* 36.4 (2017), pp. 111–122.

[Gup+17] A. Gupta, J. Johnson, A. Alahi, and L. Fei-Fei. "Characterizing and Improving Stability in Neural Style Transfer". In: *Proceedings of IEEE International Conference on Computer Vision*. 2017, pp. 4087–4096.

[HaC+11] Y. HaCohen, E. Shechtman, D. B. Goldman, and D. Lischinski. "Non-rigid dense correspondence with applications for image enhancement". In: *ACM Transactions on Graphics* 30.4 (2011), p. 70.

[Hae+07] W. V. Haevre, T. V. Laerhoven, F. D. Fiore, and F. V. Reeth. "From Dust Till Drawn: A real-time bidirectional pastel simulation". In: *The Visual Computer* 23.9–11 (2007), pp. 925–934.

[Han+08] C. Han, E. Risser, R. Ramamoorthi, and E. Grinspun. "Multiscale texture synthesis". In: *ACM Transactions on Graphics* 27.3 (2008), p. 51.

[HB17] X. Huang and S. J. Belongie. "Arbitrary Style Transfer in Real-Time with Adaptive Instance Normalization". In: *Proceedings of IEEE International Conference on Computer Vision* (2017), pp. 1510–1519.

[HE04] J. Hays and I. A. Essa. "Image and Video Based Painterly Animation". In: *Proceedings of International Symposium on Non-Photorealistic Animation and Rendering*. 2004, pp. 113–120.

[Her+01] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. "Image Analogies". In: *SIGGRAPH Conference Proceedings*. 2001, pp. 327–340.

[Her98] A. Hertzmann. "Painterly Rendering with Curved Brush Strokes of Multiple Sizes". In: *SIGGRAPH Conference Proceedings*. 1998, pp. 453–460.

[HN18] E. Heitz and F. Neyret. "High-Performance By-Example Noise Using a Histogram-Preserving Blending Operator". In: *Proceedings of the ACM SIGGRAPH / Eurographics Conference on High-Performance Graphics* 1.2 (2018), p. 31.

[Iso+17] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. “Image-to-Image Translation with Conditional Adversarial Networks”. In: 2017, pp. 5967–5976.

[JAFF16] J. Johnson, A. Alahi, and L. Fei-Fei. “Perceptual Losses for Real-Time Style Transfer and Super-Resolution”. In: *Proceedings of European Conference on Computer Vision*. 2016, pp. 694–711.

[Jam+15] O. Jamriška, J. Fišer, P. Asente, J. Lu, E. Shechtman, and D. Sýkora. “LazyFluids: Appearance Transfer for Fluid Animations”. In: *ACM Transactions on Graphics* 34.4 (2015), p. 92.

[Kaj86] J. T. Kajiya. “The Rendering Equation”. In: *SIGGRAPH Computer Graphics* 20.4 (1986), pp. 143–150.

[Kas+15] A. Kaspar, B. Neubert, D. Lischinski, M. Pauly, and J. Kopf. “Self Tuning Texture Optimization”. In: *Computer Graphics Forum* 34.2 (2015), pp. 349–360.

[Kot+19a] D. Kotovenko, A. Sanakoyeu, S. Lang, and B. Ommer. “Content and Style Disentanglement for Artistic Style Transfer”. In: *Proceedings of IEEE International Conference on Computer Vision*. 2019, pp. 4421–4430.

[Kot+19b] D. Kotovenko, A. Sanakoyeu, P. Ma, S. Lang, and B. Ommer. “A Content Transformation Block for Image Style Transfer”. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10032–10041.

[KSS19] N. I. Kolkin, J. Salavon, and G. Shakhnarovich. “Style Transfer by Relaxed Optimal Transport and Self-Similarity”. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10051–10060.

[Kwa+03] V. Kwatra, A. Schödl, I. A. Essa, G. Turk, and A. F. Bobick. “Graphcut Textures: Image And Video Synthesis Using Graph Cuts”. In: *ACM Transactions on Graphics* 22.3 (2003), pp. 277–286.

[Kwa+05] V. Kwatra, I. A. Essa, A. F. Bobick, and N. Kwatra. “Texture optimization for example-based synthesis”. In: *ACM Transactions on Graphics* 24.3 (2005), pp. 795–802.

[Kyp+13] J. E. Kyprianidis, J. Collomosse, T. Wang, and T. Isenberg. “State of the “Art”: A Taxonomy of Artistic Stylization Techniques for Images and Video”. In: *IEEE Transactions on Visualization and Computer Graphics* 19.5 (2013), pp. 866–885.

[Lai+18] W. Lai, J. Huang, O. Wang, E. Shechtman, E. Yumer, and M. Yang. “Learning Blind Video Temporal Consistency”. In: *Proceedings of European Conference on Computer Vision*. 2018, pp. 179–195.

[LH05] S. Lefebvre and H. Hoppe. “Parallel controllable texture synthesis”. In: *ACM Transactions on Graphics* 24.3 (2005), pp. 777–786.

[LH06] S. Lefebvre and H. Hoppe. “Appearance-space Texture Synthesis”. In: *ACM Transactions on Graphics* 25.3 (2006), pp. 541–548.

[Li+17] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. “Universal Style Transfer via Feature Transforms”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 385–395.

[Li+19] X. Li, S. Liu, S. D. Mello, X. Wang, J. Kautz, and M.-H. Yang. “Joint-task Self-supervised Learning for Temporal Correspondence”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 317–327.

[Lia+01] L. Liang, C. Liu, Y.-Q. Xu, B. Guo, and H.-Y. Shum. “Real-time Texture Synthesis by Patch-Based Sampling”. In: *ACM Transactions on Graphics* 20.3 (2001), pp. 127–150.

[Lia+14] J. Liao, R. Lima, D. Nehab, H. Hoppe, P. Sander, and J. Yu. “Automating Image Morphing Using Structural Similarity on a Halfway Domain”. In: *ACM Transactions on Graphics* 33.5 (2014), p. 168.

[Lia+17] J. Liao, Y. Yao, L. Yuan, G. Hua, and S. B. Kang. “Visual Attribute Transfer Through Deep Image Analogy”. In: *ACM Transactions on Graphics* 36.4 (2017), p. 120.

[Lit97] P. Litwinowicz. “Processing Images and Video for an Impressionist Effect”. In: 1997, pp. 407–414.

[Liu+18] R. Liu, J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, and J. Yosinski. “An intriguing failing of convolutional neural networks and the CoordConv solution”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 9628–9639.

[Liu+19] M.-Y. Liu, X. Huang, A. Mallya, T. Karras, T. Aila, J. Lehtinen, and J. Kautz. “Few-Shot Unsupervised Image-to-Image Translation”. In: *Proceedings of IEEE International Conference on Computer Vision*. 2019, pp. 10551–10560.

[LK81] B. D. Lucas and T. Kanade. “An iterative image registration technique with an application to stereo vision”. In: *Proceedings of International Joint Conference on Artificial Intelligence*. 1981, pp. 674–679.

[Lu+13] J. Lu, C. Barnes, S. DiVerdi, and A. Finkelstein. “RealBrush: painting with examples of physical media”. In: *ACM Transactions on Graphics* 32.4 (2013), p. 117.

[Lu+17] M. Lu, H. Zhao, A. Yao, F. Xu, Y. Chen, and X. Lin. “Decoder Network over Lightweight Reconstructed Feature for Fast Semantic Style Transfer”. In: *Proceedings of IEEE International Conference on Computer Vision* (2017), pp. 2488–2496.

[LW16a] C. Li and M. Wand. “Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis”. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2479–2486.

[LW16b] C. Li and M. Wand. “Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis”. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2479–2486.

[LW16c] C. Li and M. Wand. “Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks”. In: *Proceedings of European Conference on Computer Vision*. 2016, pp. 702–716.

[LXJ12] C. Lu, L. Xu, and J. Jia. “Combining sketch and tone for pencil drawing production”. In: *Proceedings of International Symposium on Non-Photorealistic Animation and Rendering.* 2012, pp. 65–73.

[LYT11] C. Liu, J. Yuen, and A. Torralba. “SIFT Flow: Dense Correspondence across Scenes and Its Applications”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.5 (2011), pp. 978–994.

[Mag+15] S. Magnenat et al. “Live Texturing of Augmented Reality Characters from Colored Drawings”. In: *IEEE Transactions on Visualization and Computer Graphics* 21.11 (2015), pp. 1201–1210.

[Mil+19] B. Mildenhall, P. P. Srinivasan, R. O. Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar. “Local light field fusion: Practical view synthesis with prescriptive sampling guidelines”. In: *ACM Transactions on Graphics* 38.4 (2019), p. 29.

[Mon+18] S. E. Montesdeoca, H. S. Seah, A. Semmo, P. Bénard, R. Vergne, J. Thollot, and D. Benvenuti. “MNPR: A Framework for Real-Time Expressive Non-Photorealistic Rendering of 3D Computer Graphics”. In: *Proceedings of The Joint Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering.* 2018, p. 11.

[Mul+19] R. T. Mullapudi, S. Chen, K. Zhang, D. Ramanan, and K. Fatahalian. “Online Model Distillation for Efficient Video Inference”. In: *Proceedings of IEEE International Conference on Computer Vision.* 2019, pp. 3572–3581.

[ODO16] A. Odena, V. Dumoulin, and C. Olah. “Deconvolution and Checkerboard Artifacts”. In: *Distill* (2016).

[Orz+08] A. Orzan, A. Bousseau, H. Winnemöller, P. Barla, J. Thollot, and D. Salesin. “Diffusion Curves: A Vector Representation for Smooth-Shaded Images”. In: *ACM Transactions on Graphics* 27.3 (2008), p. 92.

[PFH00] E. Praun, A. Finkelstein, and H. Hoppe. “Lapped textures”. In: *SIGGRAPH Conference Proceedings.* 2000, pp. 465–470.

[PKVP09] Y. Pritch, E. Kav-Venaki, and S. Peleg. “Shift-Map Image Editing”. In: *Proceedings of IEEE International Conference on Computer Vision.* 2009, pp. 151–158.

[Pra+01] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein. “Real-Time Hatching”. In: 2001, pp. 581–586.

[PS00] J. Portilla and E. P. Simoncelli. “A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients”. In: *International Journal of Computer Vision* 40.1 (2000), pp. 49–70.

[RDB18] M. Ruder, A. Dosovitskiy, and T. Brox. “Artistic Style Transfer for Videos and Spherical Images”. In: *International Journal of Computer Vision* 126.11 (2018), pp. 1199–1219.

[Rei+19] M. Reimann, M. Klingbeil, S. Pasewaldt, A. Semmo, M. Trapp, and J. Döllner. “Locally controllable neural style transfer on mobile devices”. In: *The Visual Computer* (2019), pp. 1–17.

[Rem+14] K. Rematas, T. Ritschel, M. Fritz, and T. Tuytelaars. “Image-Based Synthesis and Re-synthesis of Viewpoints Guided by 3D Models”. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition.* 2014, pp. 3898–3905.

[RSK10] R. Ruiters, R. Schnabel, and R. Klein. “Patch-Based Texture Interpolation”. In: *Computer Graphics Forum* 29.4 (2010), pp. 1421–1429.

[Sal+97] M. P. Salisbury, M. T. Wong, J. F. Hughes, and D. H. Salesin. “Orientable Textures for Image-based Pen-and-ink Illustration”. In: *SIGGRAPH Conference Proceedings.* 1997, pp. 401–406.

[San+18] A. Sanakoyeu, D. Kotovenko, S. Lang, and B. Ommer. “A Style-Aware Content Loss for Real-Time HD Style Transfer”. In: *Proceedings of European Conference on Computer Vision.* 2018, pp. 715–731.

[Sch+11] J. Schmid, M. S. Senn, M. Gross, and R. W. Sumner. “OverCoat: an implicit canvas for 3D painting”. In: *ACM Transactions on Graphics* 30.4 (2011), p. 28.

[SCI18] A. Shocher, N. Cohen, and M. Irani. “”Zero-Shot” Super-Resolution Using Deep Internal Learning”. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition.* 2018, pp. 3118–3126.

[SDC09] D. Sýkora, J. Dingliana, and S. Collins. “As-rigid-as-possible Image Registration for Hand-drawn Cartoon Animations”. In: *Proceedings of International Symposium on Non-Photorealistic Animation and Rendering.* 2009, pp. 25–33.

[SDM19] T. R. Shaham, T. Dekel, and T. Michaeli. “SinGAN: Learning a Generative Model From a Single Natural Image”. In: *Proceedings of IEEE International Conference on Computer Vision.* 2019, pp. 4570–4580.

[SED16] A. Selim, M. Elgharib, and L. Doyle. “Painting Style Transfer for Head Portraits Using Convolutional Neural Networks”. In: *ACM Transactions on Graphics* 35.4 (2016), p. 129.

[She+10] E. Shechtman, A. Rav-Acha, M. Irani, and S. M. Seitz. “Regenerative Morphing”. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition.* 2010, pp. 615–622.

[Shi+14] Y.-C. Shih, S. Paris, C. Barnes, W. T. Freeman, and F. Durand. “Style Transfer for Headshot Portraits”. In: *ACM Transactions on Graphics* 33.4 (2014), p. 148.

[Sho+19] A. Shocher, S. Bagon, P. Isola, and M. Irani. “InGAN: Capturing and Remapping the ”DNA” of a Natural Image”. In: *Proceedings of IEEE International Conference on Computer Vision.* 2019, pp. 4492–4501.

[SID17] A. Semmo, T. Isenberg, and J. Döllner. “Neural Style Transfer: A Paradigm Shift for Image-Based Artistic Rendering?” In: *Proceedings of International Symposium on Non-Photorealistic Animation and Rendering.* 2017, p. 5.

[Slo+01] P.-P. J. Sloan, W. Martin, A. Gooch, and B. Gooch. “The Lit Sphere: A Model for Capturing NPR Shading from Art”. In: *Proceedings of Graphics Interface*. 2001, pp. 143–150.

[SMW06] S. Schaefer, T. McPhail, and J. Warren. “Image Deformation Using Moving Least Squares”. In: *ACM Transactions on Graphics* 25.3 (2006), pp. 533–540.

[Sna+06] N. Snavely, C. L. Zitnick, S. B. Kang, and M. F. Cohen. “Stylizing 2.5-D video”. In: *Proceedings of International Symposium on Non-Photorealistic Animation and Rendering*. 2006, pp. 63–69.

[Sri+19] P. P. Srinivasan, R. Tucker, J. T. Barron, R. Ramamoorthi, R. Ng, and N. Snavely. “Pushing the Boundaries of View Extrapolation With Multiplane Images”. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 175–184.

[SZ14] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *CoRR* abs/1409.1556 (2014).

[TAY13] H. Todo, K. Anjyo, and S. Yokoyama. “Lit-Sphere Extension for Artistic Rendering”. In: *The Visual Computer* 29.6–8 (2013), pp. 473–480.

[TP91] M. Turk and A. Pentland. “Eigenfaces for Recognition”. In: *Journal of Cognitive Neuroscience* 3.1 (1991), pp. 71–86.

[Tul+18] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz. “MoCoGAN: Decomposing Motion and Content for Video Generation”. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1526–1535.

[Uly+16a] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. “Texture Networks: Feed-forward Synthesis of Textures and Stylized Images”. In: *Proceedings of International Conference on International Conference on Machine Learning*. 2016, pp. 1349–1357.

[Uly+16b] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky. “Texture Networks: Feed-Forward Synthesis of Textures and Stylized Images”. In: *ICML*. Vol. 48. 2016, pp. 1349–1357.

[UVL16] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky. “Instance Normalization: The Missing Ingredient for Fast Stylization”. In: *CoRR* abs/1607.08022 (2016).

[UVL17] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky. “Improved Texture Networks: Maximizing Quality and Diversity in Feed-Forward Stylization and Texture Synthesis”. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 4105–4113.

[Wan+17] X. Wang, G. Oxholm, D. Zhang, and Y.-F. Wang. “Multimodal Transfer: A Hierarchical Deep Convolutional Neural Network for Fast Artistic Style Transfer”. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 7178–7186.

[Wan+18a] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, and B. Catanzaro. “Video-to-Video Synthesis”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 1144–1156.

[Wan+18b] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. "High-Resolution Image Synthesis and Semantic Manipulation With Conditional GANs". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8798–8807.

[Wan+19a] M. Wang, G.-Y. Yang, R. Li, R. Liang, S.-H. Zhang, P. M. Hall, and S.-M. Hu. "Example-Guided Style-Consistent Image Synthesis From Semantic Labeling". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1495–1504.

[Wan+19b] T.-C. Wang, M.-Y. Liu, A. Tao, G. Liu, J. Kautz, and B. Catanzaro. "Few-shot Video-to-Video Synthesis". In: *Advances in Neural Information Processing Systems*. 2019, pp. 5014–5025.

[WJE19] X. Wang, A. Jabri, and A. A. Efros. "Learning Correspondence From the Cycle-Consistency of Time". In: 2019, pp. 2561–2571.

[WRB17] P. Wilmot, E. Risser, and C. Barnes. "Stable and Controllable Neural Texture Synthesis and Style Transfer Using Histogram Losses". In: *CoRR* abs/1701.08893 (2017).

[WSI07] Y. Wexler, E. Shechtman, and M. Irani. "Space-Time Completion of Video". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.3 (2007), pp. 463–476.

[Xu+11] L. Xu, C. Lu, Y. Xu, and J. Jia. "Image Smoothing via L0 Gradient Minimization". In: *ACM Transactions on Graphics* 30.6 (2011), p. 174.

[Xu+20] H. Xu, K.-H. Hui, C.-W. Fu, and H. Zhang. "TilinGNN: Learning to Tile with Self-Supervised Graph Neural Network". In: *ACM Transactions on Graphics* 39.4 (2020).

[Yüc+12] K. Yücer, A. Jacobson, A. Hornung, and O. Sorkine. "Transfusive image manipulation". In: *ACM Transactions on Graphics* 31.6 (2012), p. 176.

[Zha+19] Y. Zhang, C. Fang, Y. Wang, Z. Wang, Z. Lin, Y. Fu, and J. Yang. "Multimodal Style Transfer via Graph Cuts". In: (2019).

[Zhe+17] M. Zheng, A. Milliez, M. H. Gross, and R. W. Sumner. "Example-Based Brushes for Coherent Stylized Renderings". In: *Proceedings of International Symposium on Non-Photorealistic Animation and Rendering*. 2017, p. 3.

[Zho+17] Y. Zhou, H. Shi, D. Lischinski, M. Gong, J. Kopf, and H. Huang. "Analysis and Controlled Synthesis of Inhomogeneous Textures". In: *Computer Graphics Forum* 36.2 (2017), pp. 199–212.

[Zho+18] Y. Zhou, Z. Zhu, X. Bai, D. Lischinski, D. Cohen-Or, and H. Huang. "Non-stationary texture synthesis by adversarial expansion". In: *ACM Transactions on Graphics* 37.4 (2018), p. 49.

[Zhu+17a] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks". In: *Proceedings of IEEE International Conference on Computer Vision*. 2017, pp. 2242–2251.

[Zhu+17b] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman. “Toward Multimodal Image-to-Image Translation”. In: 2017, pp. 465–476.

[ZZ11] M. Zhao and S.-C. Zhu. “Portrait Painting Using Active Templates”. In: *Proceedings of International Symposium on Non-Photorealistic Animation and Rendering*. 2011, pp. 117–124.

Author's References

[Hau+20] F. Hauptfleisch, O. Texler, A. Texler, J. Křivánek, and D. Sýkora. "StyleProp: Real-time Example-based Stylization of 3D Models". In: *Computer Graphics Forum* 39.7 (2020), pp. 575–586.

[Jam+19] O. Jamriška, Š. Sochorová, O. Texler, M. Lukáč, J. Fišer, J. Lu, E. Shechtman, and D. Sýkora. "Stylizing Video by Example". In: *ACM Transactions on Graphics* 38.4 (2019), p. 107.

[Sýk+19] D. Sýkora, O. Jamriška, O. Texler, J. Fišer, M. Lukáč, J. Lu, and E. Shechtman. "StyleBlit: Fast Example-Based Stylization with Local Guidance". In: *Computer Graphics Forum* 38.2 (2019), pp. 83–91.

[Tex+19] O. Texler, J. Fišer, M. Lukáč, J. Lu, E. Shechtman, and D. Sýkora. "Enhancing Neural Style Transfer using Patch-Based Synthesis". In: *Proceedings of the 8th ACM/EG Expressive Symposium*. 2019, pp. 43–50.

[Tex+20a] O. Texler, D. Futschik, J. Fišer, M. Lukáč, J. Lu, E. Shechtman, and D. Sýkora. "Arbitrary Style Transfer Using Neurally-Guided Patch-Based Synthesis". In: *Computers & Graphics* 87 (2020), pp. 62–71.

[Tex+20b] O. Texler, D. Futschik, M. Kučera, O. Jamriška, Š. Sochorová, M. Chai, S. Tulyakov, and D. Sýkora. "Interactive Video Stylization Using Few-Shot Patch-Based Training". In: *ACM Transactions on Graphics* 39.4 (2020), p. 73.

Appendix A

Author's Publications

Publications Related to the Thesis

In Journals with Impact Factor

The following publications were co-authored by the author of this thesis and published in impacted journals indexed by ISI. All these publications were thoroughly presented earlier in this thesis.

O. Texler, D. Futschik, J. Fišer, M. Lukáč, J. Lu, E. Shechtman, and D. Sýkora. "Arbitrary Style Transfer Using Neurally-Guided Patch-Based Synthesis". In: *Computers & Graphics* 87 (2020), pp. 62–71 (IF = 1.351)

O. Jamriška, Š. Sochorová, O. Texler, M. Lukáč, J. Fišer, J. Lu, E. Shechtman, and D. Sýkora. "Stylizing Video by Example". In: *ACM Transactions on Graphics* 38.4 (2019), p. 107 (IF = 5.084)

Cited in:

H. Xu, K.-H. Hui, C.-W. Fu, and H. Zhang. "TilinGNN: Learning to Tile with Self-Supervised Graph Neural Network". In: *ACM Transactions on Graphics* 39.4 (2020)

M. Chu, Y. Xie, J. Mayer, L. Leal-Taixé, and N. Thuerey. "Learning Temporal Coherence via Self-Supervision for GAN-Based Video Generation". In: *ACM Transactions on Graphics* 39.4 (2020)

O. Texler, D. Futschik, M. Kučera, O. Jamriška, Š. Sochorová, M. Chai, S. Tulyakov, and D. Sýkora. "Interactive Video Stylization Using Few-Shot Patch-Based Training". In: *ACM Transactions on Graphics* 39.4 (2020), p. 73 (IF = 5.084)

Cited in:

V. Deschaintre, G. Drettakis, and A. Bousseau. "Guided Fine-Tuning for Large-Scale Material Transfer". In: *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 39.4 (2020)

D. Sýkora, O. Jamriška, O. Texler, J. Fišer, M. Lukáč, J. Lu, and E. Shechtman. "StyleBlit: Fast Example-Based Stylization with Local Guidance". In: *Computer Graphics Forum* 38.2 (2019), pp. 83–91 (IF = 2.116)

Cited in:

T. Friedrich and S. Menzel. "Standardization of Gram Matrix for Improved 3D Neural Style Transfer". In: *2019 IEEE Symposium Series on Computational Intelligence (SSCI).* 2019, pp. 1375–1382

F. Hauptfleisch, O. Texler, A. Texler, J. Křivánek, and D. Sýkora. "StyleProp: Real-time Example-based Stylization of 3D Models". In: *Computer Graphics Forum* 39.7 (2020), pp. 575–586 (IF = 2.116)

In Conference Proceedings

The following publications were co-authored by the author of this thesis and published in conference proceedings.

O. Texler, J. Fišer, M. Lukáč, J. Lu, E. Shechtman, and D. Sýkora. "Enhancing Neural Style Transfer using Patch-Based Synthesis". In: *Proceedings of the 8th ACM/EG Expressive Symposium.* 2019, pp. 43–50

Appendix B

Authorship Contribution Statement

This statement describes the specific contributions of the author of this thesis to the publications presented therein.

Arbitrary Style Transfer Using Neurally-Guided Patch-Based Synthesis (50%)

As the first author of this publication, advised by senior researchers, I come up with the way to combine patch-based and neural-based style transfer approaches. I wrote the code and later integrated a prototype into Adobe Photoshop as a plugin; I created most of the results, wrote the majority of the manuscript, and presented this work at the conference.

Stylizing Video by Example (12%)

I contributed to this paper mainly by creating a comparison with related approaches, which was challenging as our approach presents a new scenario thus related methods cannot be directly used. I helped with the experiments related to temporal coherency and blending of sequences stylized from different keyframes. Also, I helped with the preparation of results and manuscript.

Interactive Video Stylization Using Few-Shot Patch-Based Training (30%)

As the first author of this paper, advised by senior co-authors, I helped to develop the efficient training strategy for convolutional image-to-image translation networks, which allowed to create a framework that is able to perform interactive style transfer in a setting that was not possible before. I wrote a significant portion of the code, conducted most of the experiments, and notably helped with the manuscript preparation. I presented this publication at the SIGGRAPH 2020 conference, ECCV 2020 workshop, and at Real-Time Live session at SIGGRAPH where we have received Best in Show Award.

StyleBlit: Fast Example-Based Stylization with Local Guidance (14%)

In StyleBlit, my main contribution was that I extended the original method into multi-layer approach. The stylization then runs on two layers, base and detail layer, with different parameters. This helps to hide the seams original StyleBlit is prone to produce. To reduce seams even more, I developed the post-processing step that moves the chunks in order to align them, thus remove seams. Finally, I presented this paper at the Eurographics 2019 conference.

StyleProp: Real-time Example-based Stylization of 3D Models (20%)

In StyleProp, I was co-advising throughout the entire research process, I helped to formulate the problem and propose its solution. I was actively participating on writing of the manuscript, creating comparison with related work, and preparing the results.

Appendix C

Stylizing Video by Example Supplementary Material

In this supplementary material section we first present an example of a more complex composition stylized using multiple layers (see Fig. C.1). We compare our technique with the recent neural-based style transfer approach of Sanakoyeu et al. [San+18] (Fig. C.2) and demonstrate artifacts that would appear when the stylized keyframe is only advected using optical flow (Fig. C.4). We also present an example of advecting pixel selection mask (Fig. C.3) and a detailed view on three different challenging scenarios that cause difficulties to our method (Figures C.5, C.6, and C.7). Finally, we demonstrate a sequence stylized using our method with three different artistic styles (Fig. C.8).

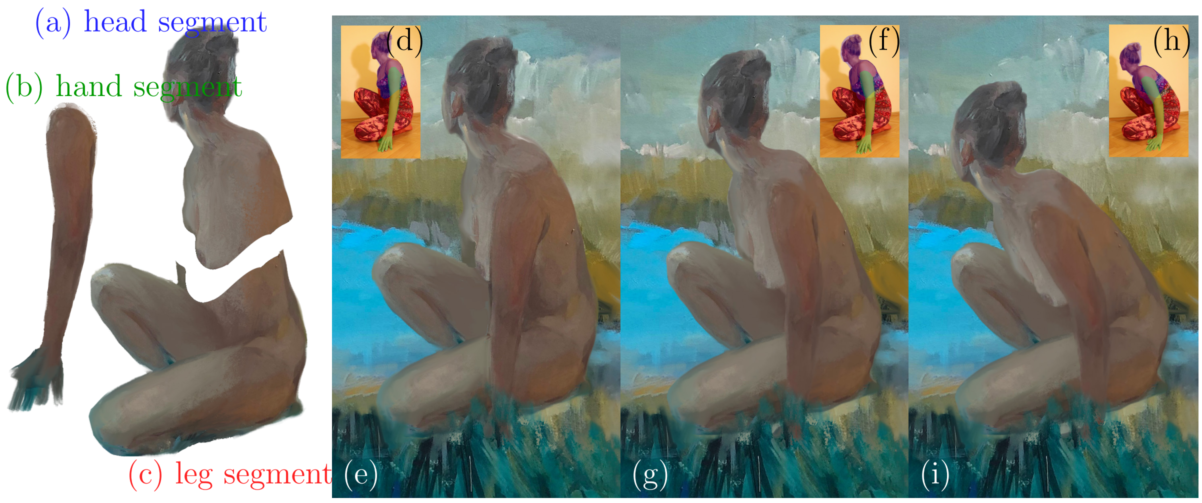


Figure C.1: *Complex composition: the style exemplar is segmented into individual components, (a) head segment, (b) hand segment and (c) leg segment. The same segmentation is performed on the target sequence (d), (f), and (h). The stylized components are composed together with the background (e), (g), and (i). Target video frames (d, f, h), style exemplars (a, b, c), and the final composition (e, g, i) courtesy of © Markéta Kolářová, used with permission.*

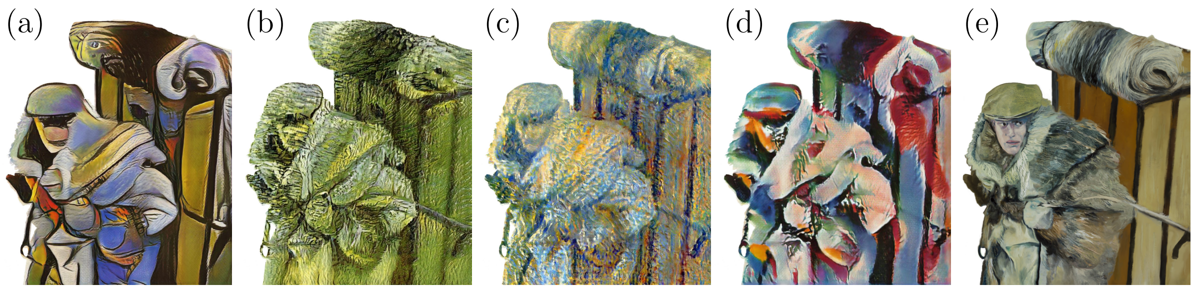


Figure C.2: *Comparison with the recent neural-based method Sanakoyeu et al. [San+18] pre-trained on various styles: (a) Picasso, (b) Van-Gogh, (c) Monet, and (d) Kandinsky. Although this method can reproduce local characteristics of the trained paintings it is unable to preserve stylization in a semantically meaningful way, i.e., stylize differently the face region and the coat, the way a real painter would. In contrast our approach (e) preserves better the style characteristics and its semantic context.*

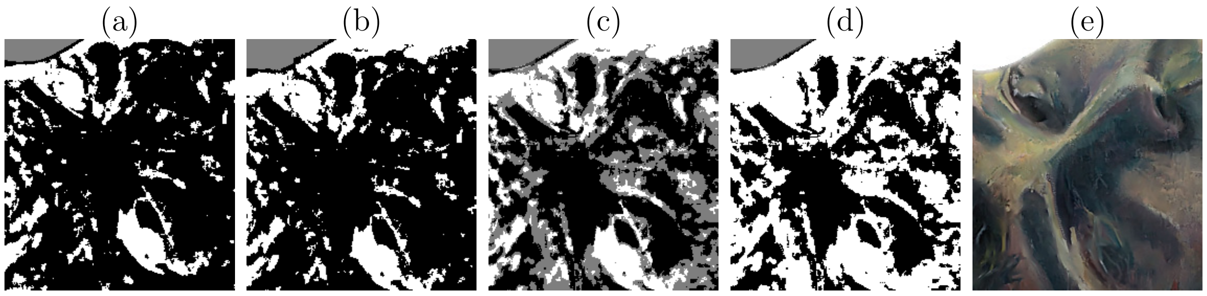


Figure C.3: *A pixel selection mask Z_{i-1} (a) is advected using inter-frame optical flow D_i to produce $\hat{Z}_i$ (b). Then a lower synthesis error constraint is applied, such that pixels that were already assigned to retrieve content from keyframe S_b (in white) remain unchanged and only pixels assigned to keyframe S_a (in black) are updated (in gray) (c) to obtain the final pixel mask Z_i (d). Z_i is then used to produce the fused frame O_i (e).*

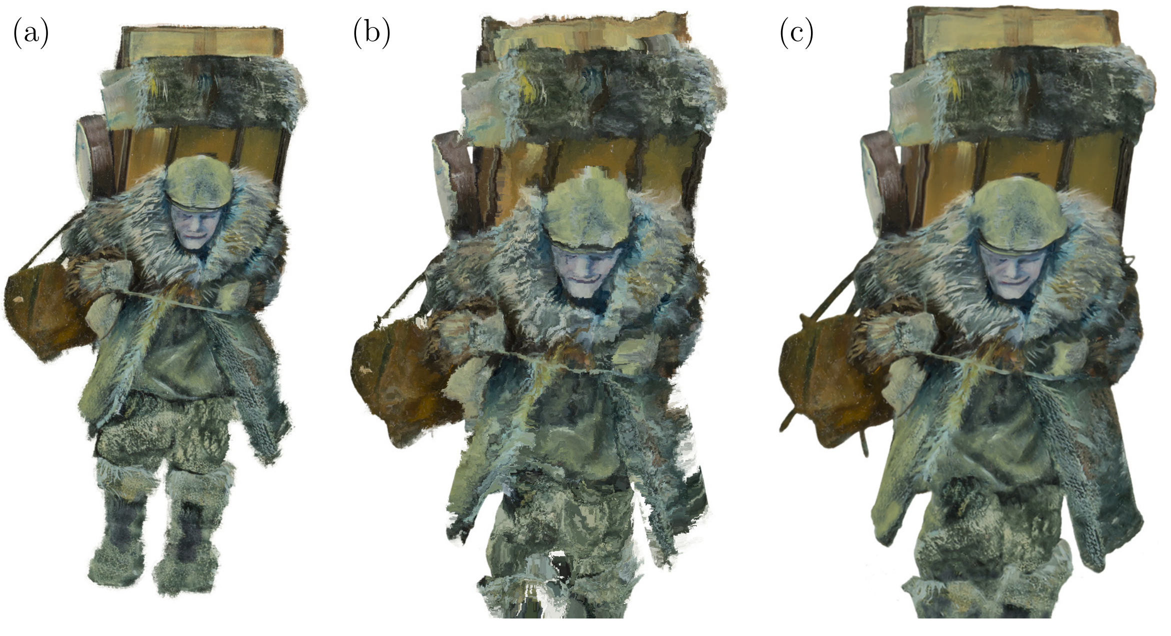


Figure C.4: *Flow advection: (a) style exemplar, (b) advection of the style exemplar using optical flow directly introduces significant deformations and distortions, (c) our result - we use advection to generate the G_{pos}, which is then used only to guide the synthesis together with the other guidance channels. Style exemplar (a) courtesy of © MAUR film, Václav Švankmajer, used with permission.*

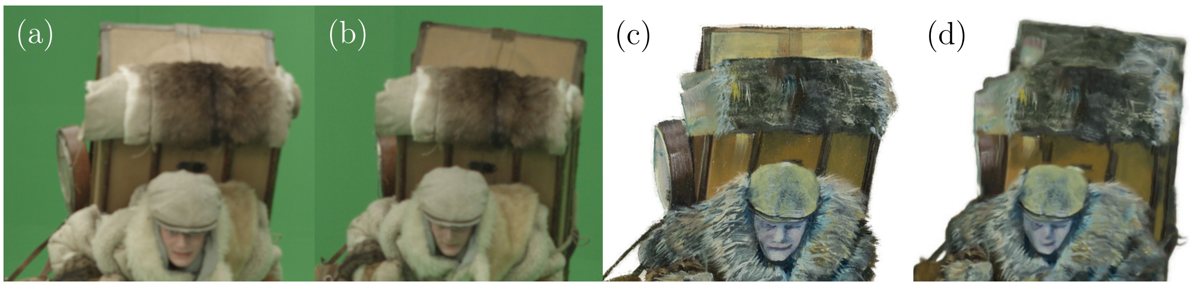


Figure C.5: *Challenge I - illumination variation: appearance variations in the target video sequence (a, b) might introduce large error in G_{col}. This in fact may result in copying style texture from the wrong places in the exemplar (c) onto the new frame (d). Note the upper part of the wooden box on subject's back, where fur texture is copied onto the wrong place. To certain extent, this problem can be suppressed using additional guiding channel, G_{edge}. In this figure G_{edge} was disabled for illustration purposes. Target video frames (a, b) and style exemplar (c) courtesy of © MAUR film, Václav Švankmajer, used with permission.*

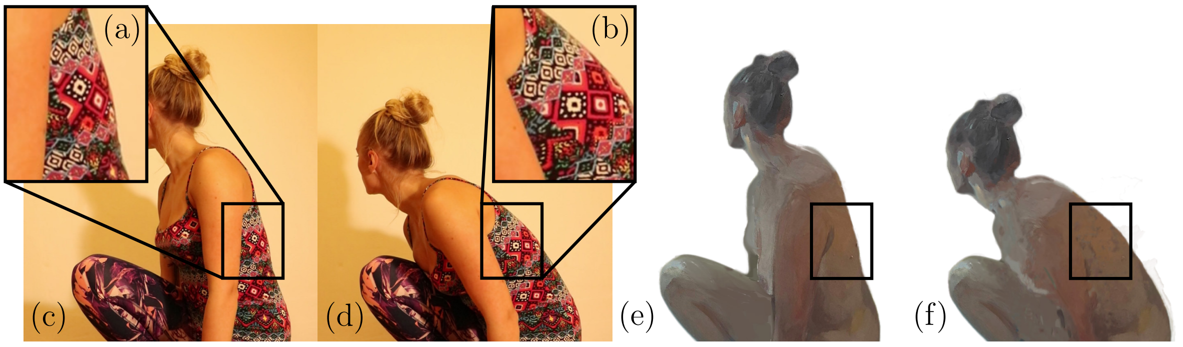


Figure C.6: *Challenge II - complex texture: when the stylized object has salient patterns or texture (a) even small change in orientation (b) might introduce very high error in G_{col} due to matching of patches with highly inconsistent content. This may result in suboptimal synthesis results such as the apparent blurring artifacts, compare (c) and (d). Target video frames (a, b) and style exemplar (c) courtesy of © Markéta Kolářová, used with permission.*

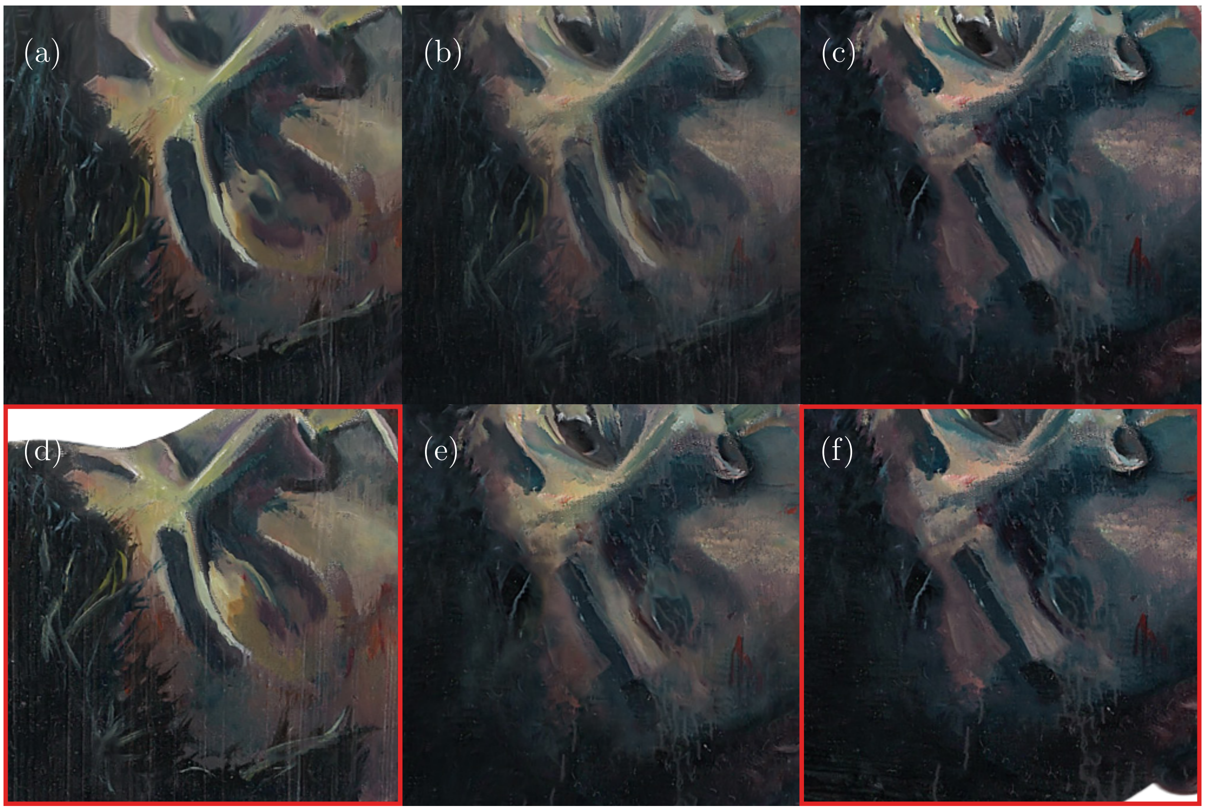


Figure C.7: *Challenge III - structural changes: frames (a) and (c) were stylized using different keyframes (d) and (f), due to this reason their features are not aligned perfectly (see eyebrows) and they also have slightly different color distribution. When linear blending is used (b) the resulting fused image suffer from apparent ghosting and contrast loss. When our gradient domain fusion with contrast-preserving blend is applied (e), the contrast is similar to the original keyframes and the ghosting on eyebrows is less apparent, however, still a bit visible. Style exemplars (d, f) courtesy of © MAUR film, Václav Švankmajer, used with permission.*

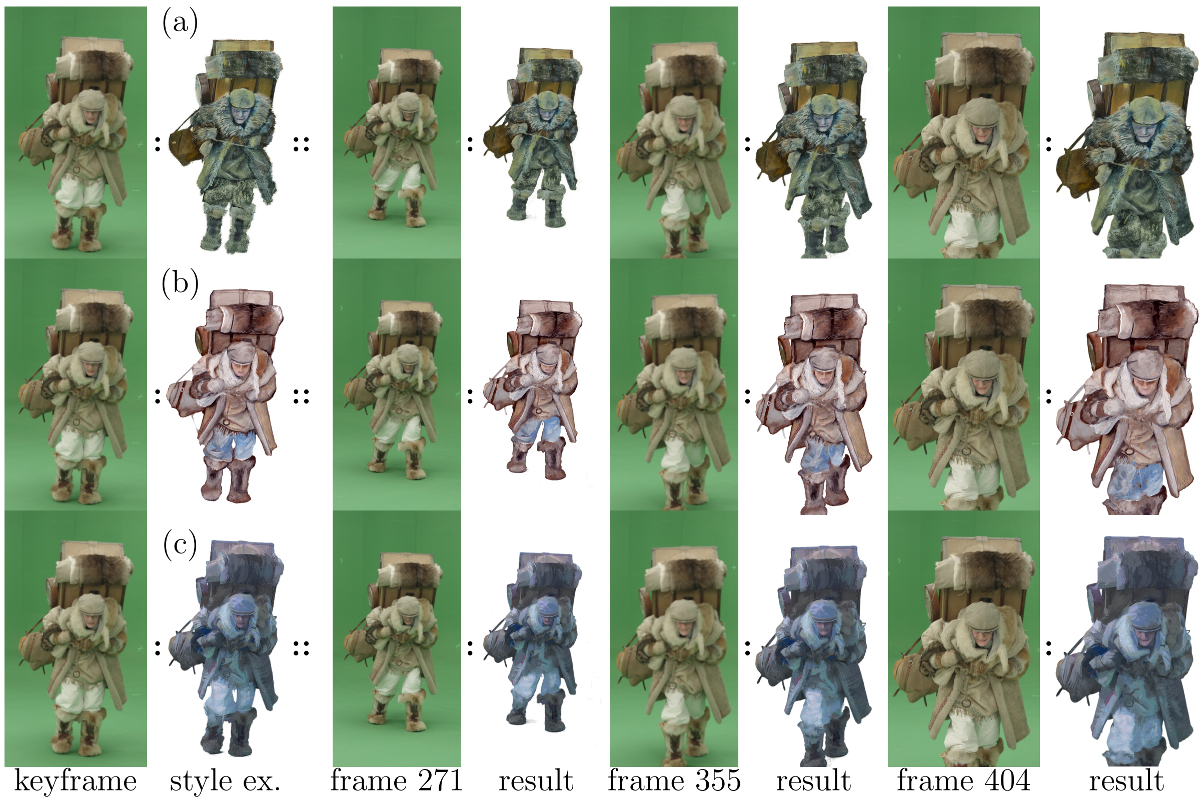


Figure C.8: *A target sequence stylized using three different styles. Target video frames and style exemplar (a) courtesy of © MAUR film, Václav Švankmajer, style exemplar (b) courtesy of © Pavla Sýkorová, style exemplar (c) courtesy of © Jakub Javora, used with permission.*

Appendix D

Patch-Based Training Supplementary Material

In this supplementary material section we describe the interactive applications of our framework in more detail, presenting the overall architecture of the solution as well as mentioning the specific hardware we used. Furthermore, we show example photos of our framework during real-time stylization sessions with artists (see our supplementary video for live recordings from those sessions) and discuss feedback we received during our informal user study. Lastly, we show additional results produced by our framework, and additional experiments with hyper-parameter setting.

D.1 Interactive applications

To demonstrate interactive applications, we provide artists with a setup of our framework in a few variations. Each scenario involves working with a workstation PC, equipped with a consumer-grade GPU (we use Nvidia RTX 2080), on which the artists perform a task. This machine runs our framework executable, which displays visual feedback for the artist. Training of the model is done off-site on a server with an Nvidia Tesla V100 GPU. The client machine sends necessary training data to this server and the training server in turn periodically sends back models trained with the new data. The training

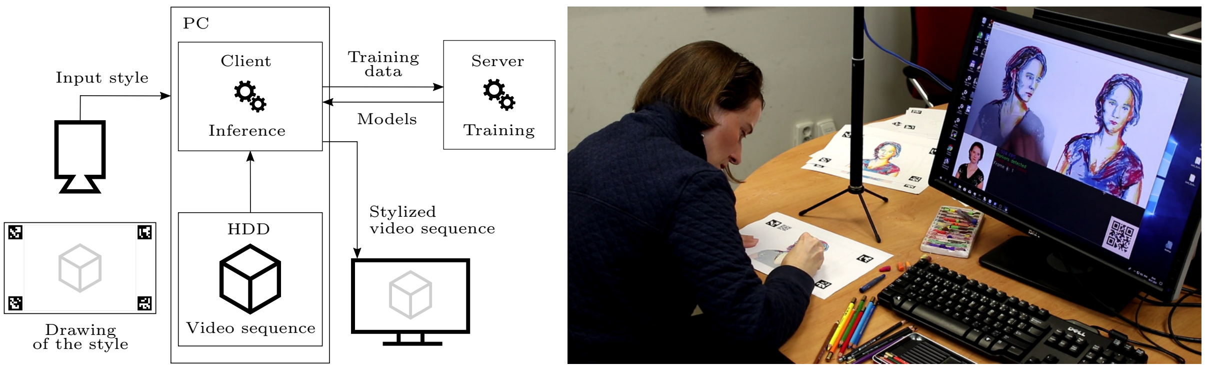


Figure D.1: *Scenario No. 1: an artist is drawing over a stencil of a keyframe using traditional media. The stencil contains markers that allow us to perfectly align the frames to prevent shift in images.*

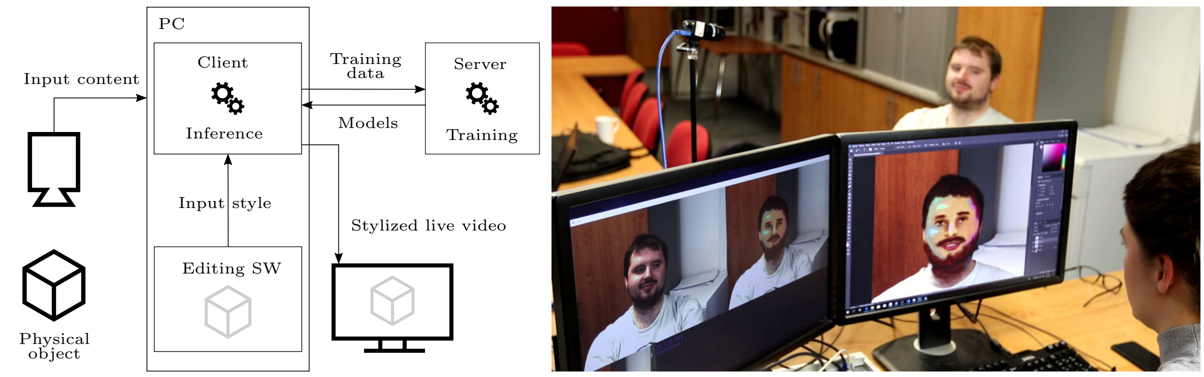


Figure D.2: *Scenario No. 2: an artist is stylizing an object as seen by the camera in real-time using image editing software.*

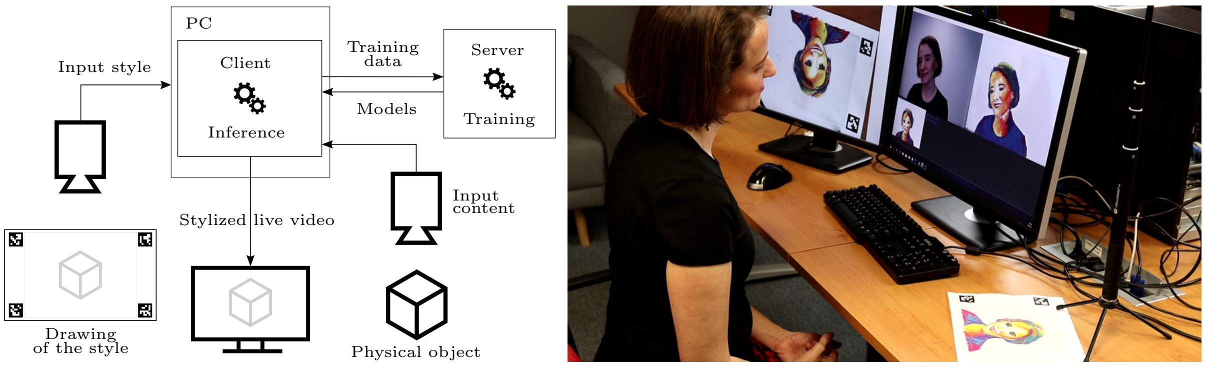


Figure D.3: *Scenario No. 3: an artist is stylizing an object as seen by the camera in real-time using a physical stencil.*

data is replaced every time the server receives a new version of a frame. Our training process quickly adapts the model to the new data.

Trained models are used on the artist's PC to generate stylized video frames. Our approach allows us to display an acceptable result in as little as 5 seconds, which improves with time as better models arrive. In practice, the potentially lengthy process of art creation amortizes training time, largely masking the downside of this delay.

Note that inference could also be performed on the server but we do it locally to reduce delay during live-feed stylization.

We devise the following real-time style transfer tasks:

Pre-recorded video + live style capture (traditional)

The artist is provided with (or creates) a pre-recorded video sequence and selects one or more keyframes which they will paint over. These keyframes are printed in low contrast on a stencil with markers. These markers allow us to perfectly match and align the contents of the stencil with the input sequence frames, so as to avoid misalignment of the training data and achieve the best performance possible. In case of multiple keyframes, we differentiate stencils using additional markers so that the artist is free to swap between them during the session.

As the artist starts painting the first keyframe, the server recognizes which keyframes are ready and only uses previously seen keyframes to train on. Unfinished or unseen parts will likely produce poor visual results which will indicate spots which need to be fixed in current or other keyframes. The artist may also wish to create masks for each keyframe, to prevent introducing ambiguity of different appearances for identical content or to save repetitive work, especially if the keyframes are relatively similar. Diagram for this setup and an example photograph are shown in Fig. D.1.

Live video capture + live style capture (digital)

This scheme is different from the previous in that there is no pre-recorded video sequence, instead, we arrange a camera, capturing a scene in real-time. Our framework allows the artist to export a still image of the scene into image editing software of their choice. This image can then be edited or painted over to achieve an artistic look. Its modified version is periodically sent to the training server, where it serves as the current style exemplar used for training.

During the session, the artist is free to change the scene, while observing the stylization in real-time. If the scene contains some object, a common modification of the scene would be rotating or moving the object. Once the artist is satisfied with the result, they can export additional still images to fix any issues in the scene. This could be, for example, one image for the front of an object and another image for the back of the object. Diagram for this scenario and an example photograph of a session are shown in Fig. D.2.

Live video capture + live style capture (traditional)

We design our framework to also let us combine the two previous scenarios. When a still image of a live scene is exported, it can be printed on a stencil. Artist draws on that stencil and we set up a second camera to capture it. The framework automatically aligns it to the still image and sends it to the training server again. Defining multiple keyframes is then as simple as printing multiple different stencils with identifying markers.

Although working with a digital image is often faster, this setup is useful due to the preference of some artists to work with traditional artistic media. Our framework is well suited for capturing real strokes and stylizing the video frames in a way similar to traditional animation. This scenario is visually explained in Fig. D.3.

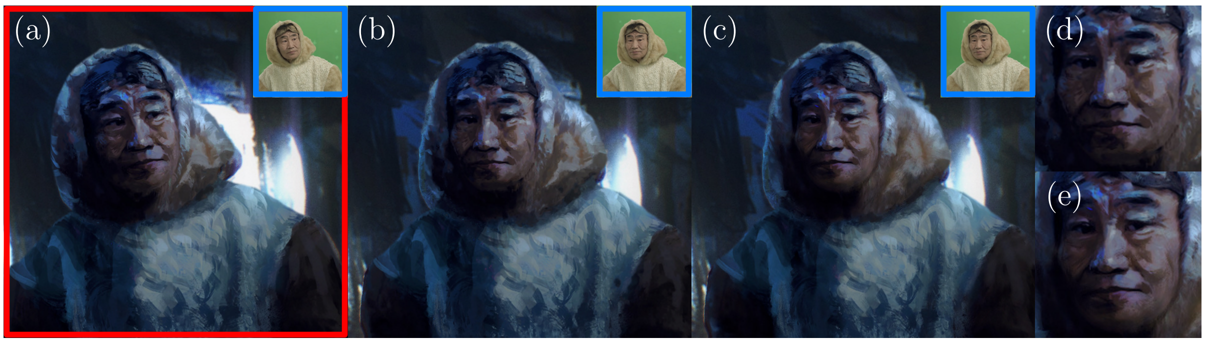


Figure D.4: *The keyframe (a) was used to produce the sequence of 148 frames. While the body part is faithfully represented in both [Jam+19] (b) and ours (c), our approach better preserves the facial region; see the zoom-in views [Jam+19] (d) and ours (e). Video frames (insets of a–c) courtesy of © MAUR film and style exemplar (a) courtesy of © Jakub Javora.*

D.2 User study

We asked the artists for their comments on using our framework. Although our user study was informal, we believe it still presents an interesting insight into the contibution of this work.

One of the very first impressions was the moment of surprise and awe whenever a new model arrived on the client machine and a better stylization started appearing on the screen. Thanks to this effect, the artists felt engaged throughout the whole session, some even asked us for further sessions so they could explore the implications of our framework more.

Generally, artists tended to describe the proposed system as a completely new tool to approaching artistic animation, thanks to the real-time feedback and continuous improvement. The other aspect that makes using our framework easy and entertaining, according to the comments, is using the photo stencils, as painting over a photograph using brushes is much easier than creating art from scratch. This also makes it suitable for children, who are largerly familiar with using stencils from coloring books.

Lastly, artists appreciated the fact that no explicit masking needs to be done during the creation process (e.g., background masking). The model we use seems good at representing identity transformation, thus leaving parts of the image unstylized means that the original background just propagates to the output.

While the overwhelming majority of the comments we received were positive, the one negative remark was that the result image quality is somewhat lower than well-optimized sequence created by Jamriška et al. [Jam+19]. However, compared to the inability of their method to deliver such a real-time experience, we feel our framework makes for a reasonable trade-off.

D.3 Additional Results and Experiments

In this section, we first present an additional result of our approach compared to the result of Jamriška et al. [Jam+19], see Fig. D.4.

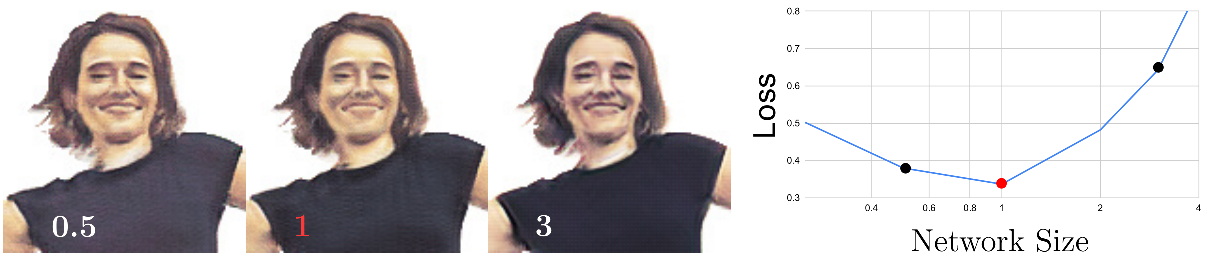


Figure D.5: *Impact of network size on the visual quality of results. The loss, y-axes, is computed w.r.t. the output of Jamriška et al. [Jam+19]. The x-axes shows the network size (i.e., number of filters) relative to the best setting we found via hyper-parameter search. Other hyper-parameters are fixed. The middle image (1) depicts the best setting, the left image (0.5) represents setting with half number of filters, and the right image (3) represents setting with three times more filters compared to the middle image. The difference in the visual quality of images, as well as the loss curve, clearly show that there exists a saddle point.*

Second, as already primarily covered in the main text, we discuss hyper-parameter optimization on one more example. As it is a common practice to reduce the network size to prevent overfitting, in Fig. D.5, we demonstrate that in the task of style transfer, certain network capacity is necessary to achieve high-quality results.

Appendix E

StyleBlit Supplementary Material

In this supplementary material section we present pseudocode of the brute-force StyleBlit algorithm (Algorithm 2), additional results of our method, and a description of perceptual study we conducted in order to evaluate visual quality of our results (see Section E.1).

In Fig. 7.6 we copmare our approach with StyLit [Fi16] and The Lit Sphere [Slo+01] in the scenario where normals are used for guidance. Additional results comparing our approach with StyLit are presented also in Figures E.1, E.2, E.3, and E.4. More results for normal-based guidance are presented in Figures E.5, E.6, and E.7 where our approach has been applied to CAD model. Fig. E.8 further compares our method with texture mapping for both scenarios presented in the main paper, i.e., when the style exemplar is drawn on a 2D projection of a 3D model and on a planar unwrap of this model. For this scenario an example of guiding channels is presented in Fig. E.10. Finally, a reference result of neural-based style transfer is demonstrated in Fig. E.12 and in Fig. E.9 additional comparison with FaceStyle [Fi17] is presented together with an example of corresponding guiding channels in Fig. E.11.

E.1 Perceptual Study

To verify the fact that our method produces results comparable to the output of StyLit algorithm [Fi16] we conducted a perceptual study where we asked 13 participants 6 men

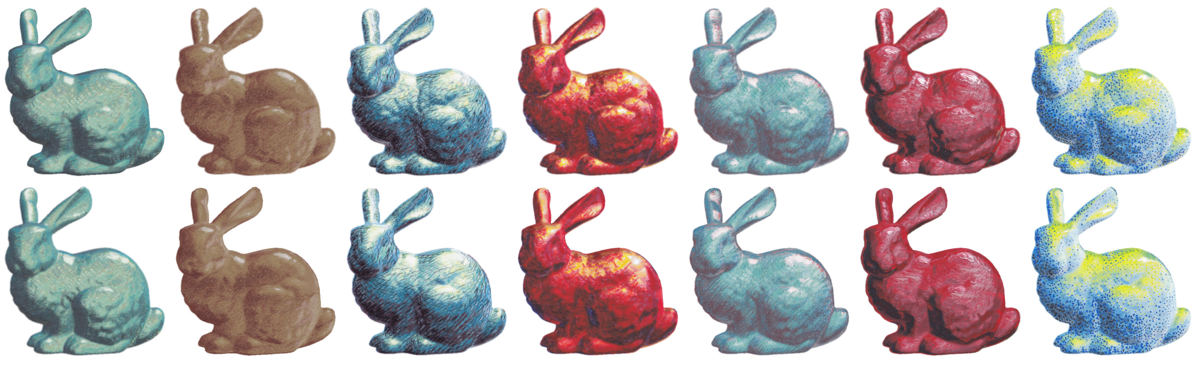


Figure E.1: *Comparison with StyLit [Fi16] (normal-based guidance): our approach (first row), StyLit (second row).*

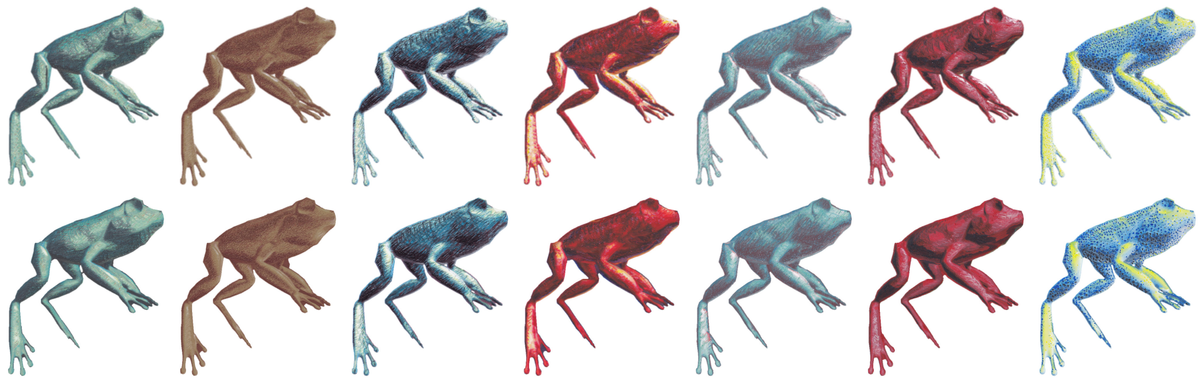


Figure E.2: *Comparison with StyLit [Fi16] (normal-based guidance): our approach (first row), StyLit (second row).*

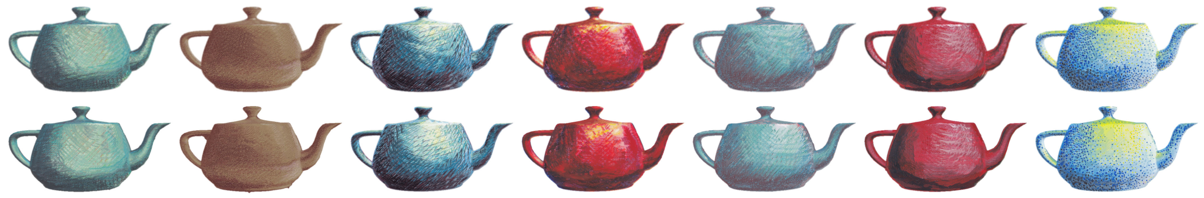


Figure E.3: *Comparison with StyLit [Fi16] (normal-based guidance): our approach (first row), StyLit (second row).*

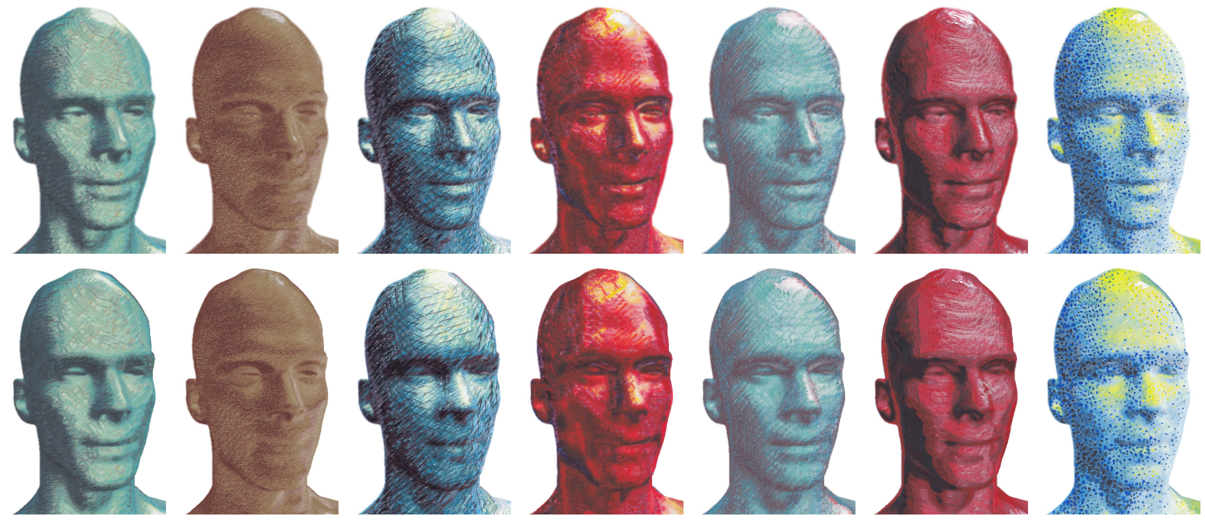


Figure E.4: *Comparison with StyLit [Fi16] (normal-based guidance): our approach (first row), StyLit (second row).*

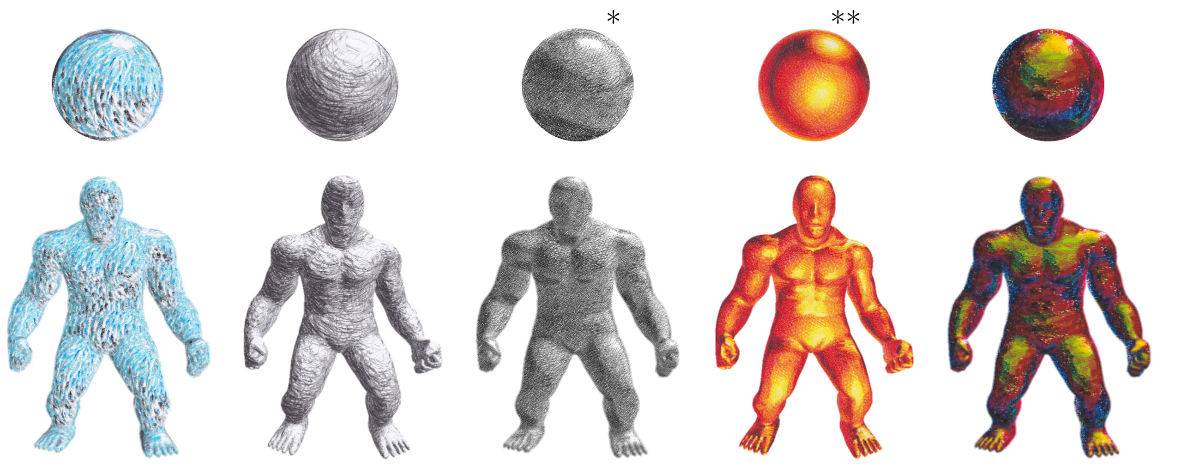


Figure E.5: *Additional results with normal-based guidance demonstrating diversity of style exemplars that can be used in our method: style exemplars (top row), result of our method (bottom row). Style exemplars:* © Pavla Sýkorová, Daichi Ito*, *and* Zuzana Studená**

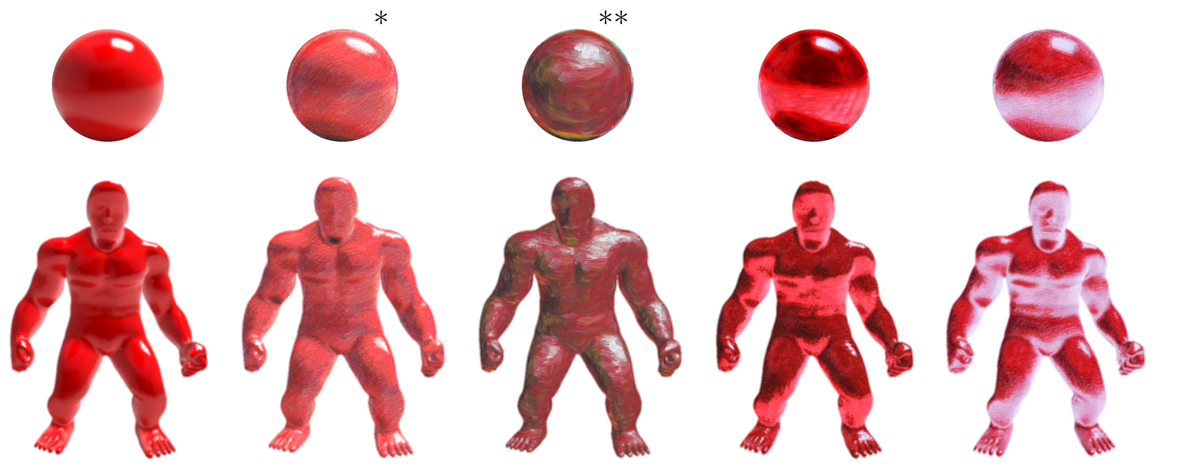


Figure E.6: *Additional results for an application where normal values are used as a local guide: source style exemplars (spheres), stylized targets (golems). Style exemplars:* © Karel Seidl, Daichi Ito*, *and* Pavla Sýkorová**

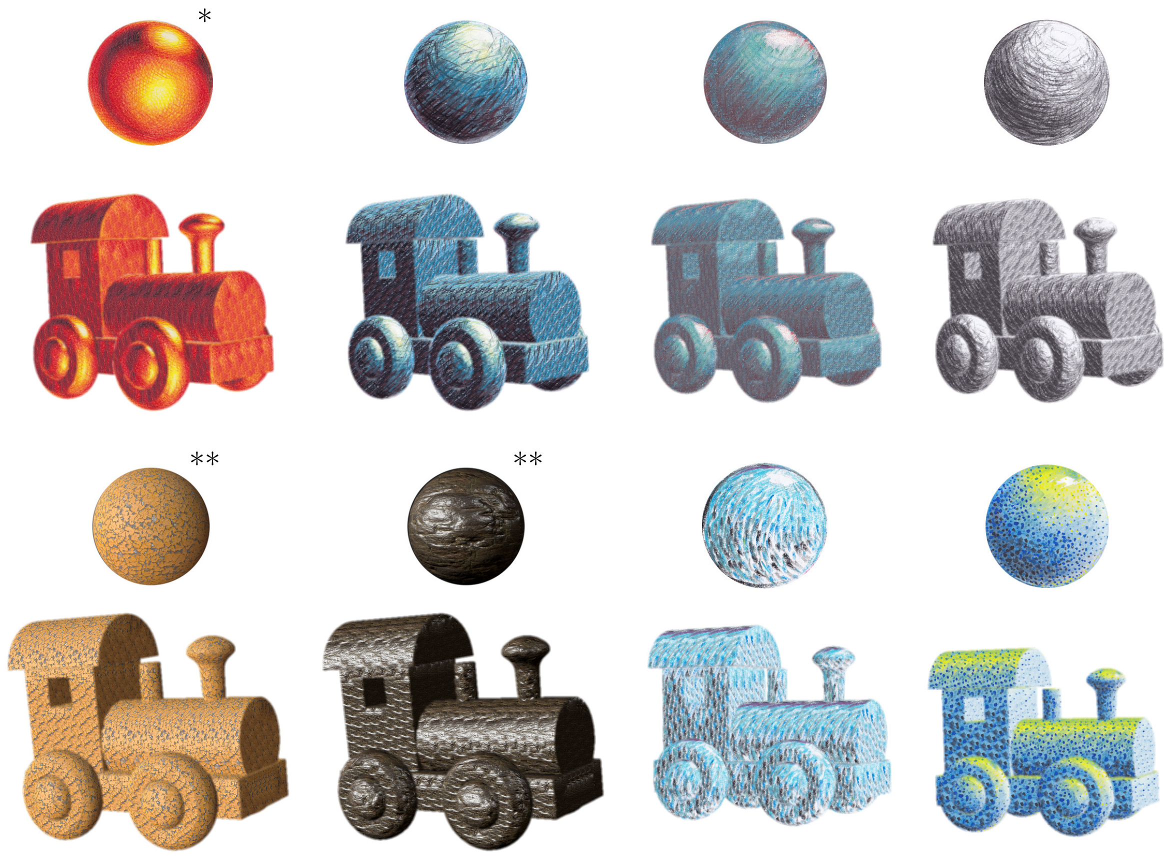


Figure E.7: *Our approach applied on a CAD model (normal-based guidance): source style exemplars (spheres), stylized targets (trains). Style exemplars:* © Pavla Sýkorová, Zuzana Studená*, *and* Free PBR**

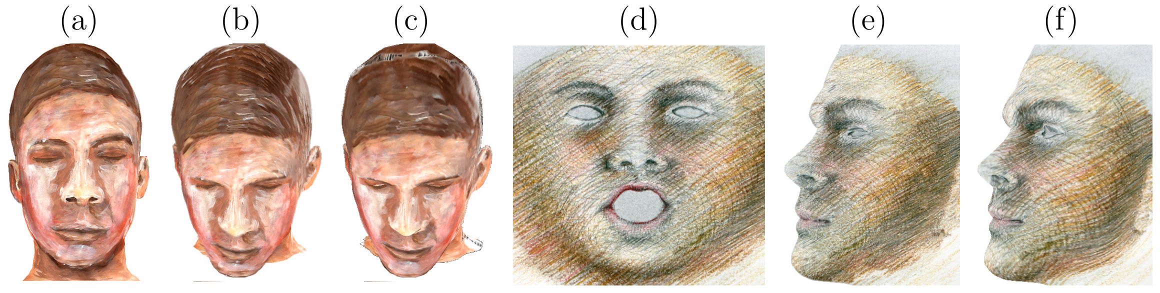


Figure E.8: *Comparison with texture mapping: style exemplar drawn on a 2D projection of a 3D model (a) and on a planar unwrap of this model (d), new viewpoint generated using our approach (b, e) and using texture mapping (c, f). Style exemplars:* © Pavla Sýkorová

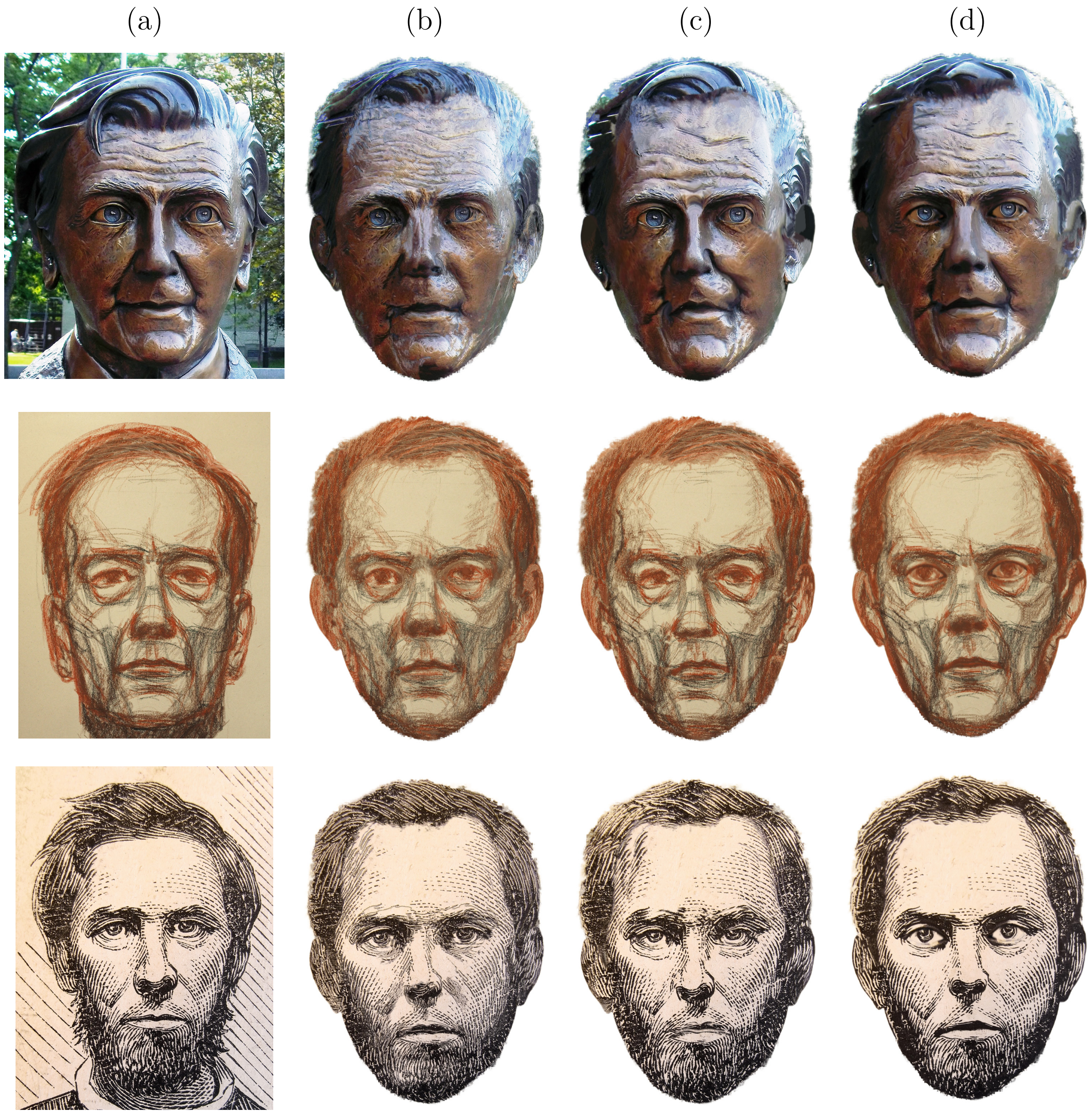


Figure E.9: *Comparison with FaceStyle [Fi17]: style exemplar (a), result of our method with strong (b) and weak (c) apperance guide, result of FaceStyle (d). Style exemplars (top to bottom):* © Léonard Simard, Adrian Morgan, *and* Thomas Shahan

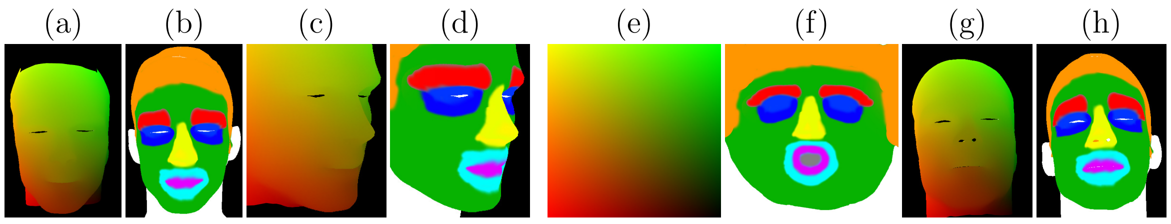


Figure E.10: *Examples of guiding channels for an application where texture coordinates are used as a local guide. Scenario where a 2D projection of a 3D model is used as an exemplar: texture coordiantes guide for source (a) and target (c), segmentation guide for source (b) and target (d). Scenario where a planar unwrap of a 3D model is used as an exemplar: texture coordiantes guide for source (e) and target (g), segmentation guide for source (f) and target (h).*

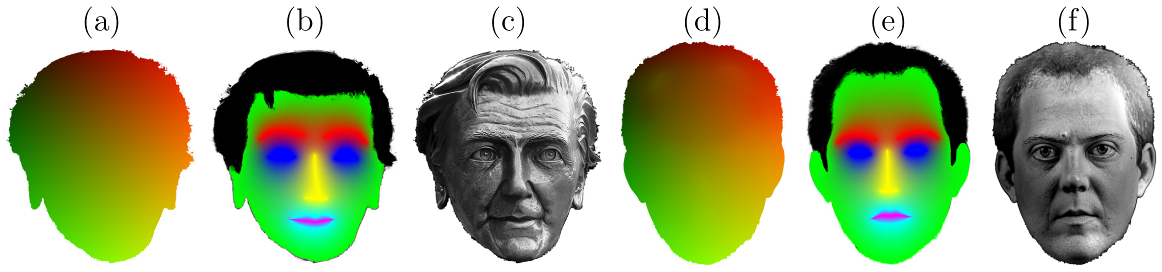


Figure E.11: *Example of guiding channels for an application where displacement field is used as a local guide (FaceStyle [Fi17]): displacement field guide for source (a) and target (d), segmentation guide for source (b) and target (e), apperance guide for source (c) and target (f).*

Algorithm 2: StyleBlit

Inputs : source style exemplar C_S, source guides G_S, target guides G_T, threshold t.
Output: target stylized image C_T.

```
StyleBlit():
    for each pixel $\boldsymbol{p} \in C_T$ do
        if $C_T[\boldsymbol{p}]$ is empty then
            $\boldsymbol{u}^\star = argmin_{\boldsymbol{u}} ||G_T[\boldsymbol{p}] - G_S[\boldsymbol{u}]||$
            for each pixel $\boldsymbol{q} \in C_S$ do
                if $C_T[\boldsymbol{p} + (\boldsymbol{q} - \boldsymbol{u}^\star)]$ is empty then
                    $e = ||G_T[\boldsymbol{p} + (\boldsymbol{q} - \boldsymbol{u}^\star)] - G_S[\boldsymbol{q}]||$
                    if $e < t$ then
                        $C_T[\boldsymbol{p} + (\boldsymbol{q} - \boldsymbol{u}^\star)] = C_S[\boldsymbol{q}]$
```

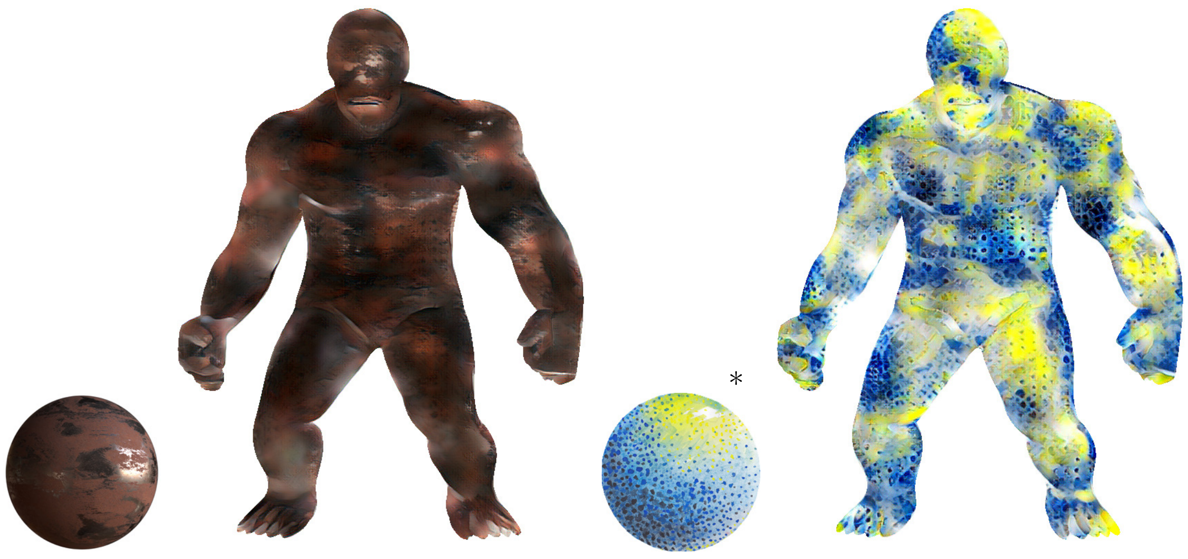


Figure E.12: *Reference result of neural-based style transfer [Li+17]: source style exemplars (spheres), stylized targets (golems). Style exemplars:* © Free PBR *and* Pavla Sýkorová*

and 7 women between in ages between 21 and 47. From this group 6 people had previous hands-on experience with art and computer graphics while the other 7 were uninformed observers. We showed the participants result produced by our approach side-by-side with the result of StyLit algorithm for different style exemplars presented in Fig. 7.6 and asked them which of the two presented renders better reproduces the original artistic style. During the experiment participants were asked to consider only the quality of artistic style, preservation of geometric details were stated as unimportant. The null hypothesis was that "there is no significant statistical difference between our approach and StyLit with respect to perceived style transfer quality". The value of χ^2-test was equal to 0.694 which clearly confirmed this null hypothesis.