DOI: 10.1111/cgf.15155

Eurographics Symposium on Rendering 2024
E. Garces and E. Haines
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 43 (2024), Number 4

Scaling Painting Style Transfer

Bruno Galerne[1,2], Lara Raad[3], José Lezama[3†], and Jean-Michel Morel[4]

[1]Institut Denis Poisson, Université d'Orléans, Université de Tours, CNRS [2]Institut Universitaire de France (IUF)
[3]Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República [4]City University of Hong Kong
[†]José Lezama is now at Google Research. Contributed to this work while at Universidad de la República.

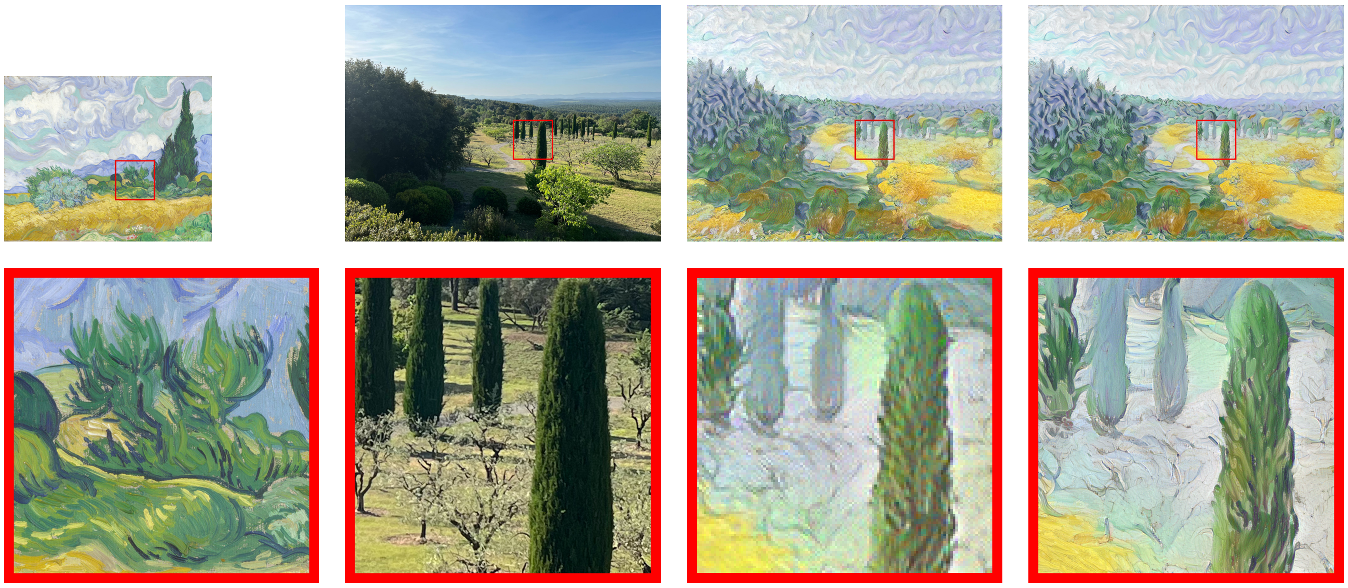

Figure 1: *Ultra-high resolution multiscale style transfer. Top row from left to right: style (4226×5319), content (6048×8064), intermediary transfer at scale 1 (756×1008) and final transfer at scale 4 (6048×8064) (intermediary results at scale 2 (1512×2016) and 3 (3024×4032) are not shown). Bottom row: zoomed in detail of size 1024×1024 for the three UHR images and 128×128 for the low resolution transfer at scale 1. Our method produces a style transfer of unmatched quality for such high resolution. It effectively conveys a pictorial aspect to the output images thanks to fine painting details such as brushstrokes, painting cracks, and canvas texture.*

Abstract

Neural style transfer (NST) is a deep learning technique that produces an unprecedentedly rich style transfer from a style image to a content image. It is particularly impressive when it comes to transferring style from a painting to an image. NST was originally achieved by solving an optimization problem to match the global statistics of the style image while preserving the local geometric features of the content image. The two main drawbacks of this original approach is that it is computationally expensive and that the resolution of the output images is limited by high GPU memory requirements. Many solutions have been proposed to both accelerate NST and produce images with larger size. However, our investigation shows that these accelerated methods all compromise the quality of the produced images in the context of painting style transfer. Indeed, transferring the style of a painting is a complex task involving features at different scales, from the color palette and compositional style to the fine brushstrokes and texture of the canvas. This paper provides a solution to solve the original global optimization for ultra-high resolution (UHR) images, enabling multiscale NST at unprecedented image sizes. This is achieved by spatially localizing the computation of each forward and backward passes through the VGG network. Extensive qualitative and quantitative comparisons, as well as a perceptual study, show that our method produces style transfer of unmatched quality for such high-resolution painting styles. By a careful comparison, we show that state-of-the-art fast methods are still prone to artifacts, thus suggesting that fast painting style transfer remains an open problem.

CCS Concepts

• ***Computing methodologies*** → ***Image manipulation;***

1. Introduction

Style transfer is an image editing strategy transferring an image style to a content image. Given style and content, the goal is to extract the style characteristics of the style and merge them to the geometric features of the content. While this problem has a long history in computer vision and computer graphics (e.g., [HJO*01; APH*14]), it has seen a remarkable development since the seminal works of Gatys *et al.* [GEB15; GEB16] that introduced *neural style transfer* (NST) [JYF*20]. These works demonstrate that the Gram matrices of the activation functions of a pre-trained VGG19 network [SZ15] faithfully encode the perceptual style and textures of an input image. NST is performed by optimizing a functional aiming at a compromise between fidelity to VGG19 features of the content image while reproducing the Gram matrices statistics of the style image. Other global statistics have been proven effective for style transfer and texture synthesis [LZW16; SC17; LPSB17; VDKC20; RWB17; HVCB21; DDD*21; GGL22] and it has been shown that a coarse-to-fine multiscale approach allows one to reproduce different levels of style detail for images of moderate to high-resolution (HR) [GEB*17; Sne17; GGL22]. The two major drawbacks of such optimization-based NST are the computation time and the limited resolution of images because of large GPU memory requirements. The former limitation is more critical for the present work, since conveying the visual aspects of a painting requires multiple scales of visual detail.

Regarding computation time, several methods have been proposed to generate new stylized images by training feed-forward networks [ULVL16; JAF16; LW16] or by training VGG encoder-decoder networks [CS16; HB17; LFY*17; LLKY19; CG20]. These models tend to provide images with relatively low style transfer loss and can therefore be considered as approximate solutions to [GEB16]. Despite a remarkable acceleration, these methods suffer from GPU memory limitations due to the large size of the models used for content and style characterization and are therefore limited in terms of resolution (generally limited to 1024^2 pixels (px)).

This resolution limitation has received less attention but was recently tackled [ALH*20; WLW*20; CWX*22; WZZ*23]. Nevertheless, although generating UHR images (larger than 4k images), the approximate results are not able to correctly represent the style resolution. Indeed, for some methods to satisfy the GPU's memory limitations, the transfer is performed locally on small patches of the content image with a zoomed out style image (1024^2 px) [CWX*22]. In other methods, the multiscale nature of the networks is not fully exploited [WLW*20].

At the opposite of these machine learning based-approaches, we propose to solve the original NST optimization problem [GEB16] for UHR images by introducing an exact localized algorithm. As illustrated in Figure 1, our UHR multiscale method manages to transfer the different levels of detail contained in the style image from the color palette and compositional style to the fine brushstrokes and canvas texture. The resulting UHR images look like authentic painting as can be seen in the UHR example of Figure 2.

Comparative experiments show that the results of competing methods suffer from brushstroke styles that do not match those of the UHR style image, and that very fine textures are not transferred well and are subject to local artifacts. To straighten this visual comparison, we also introduce a qualitative and quantitative *identity test* that highlights how well a given texture is being emulated. A perceptual study completes these experiments and confirm the superiority of our approach regarding painting style reproduction.

The main contributions of this work are summarized as follows:

- We introduce a two-step algorithm to compute the style transfer loss gradient for UHR images that do not fit in GPU memory using localized neural feature calculation.
- We show that using this algorithm in a multi-scale procedure leads to a UHR style transfer for images up to $20k^2$ px with details conveying a natural painting aspect at *every scale*.
- Comparative experiments show that the visual quality of our UHR style transfer is by far richer and more faithful than state-of-the-art fast but approximate solutions, revealing that, in our opinion, fast UHR painting style transfer is still an open problem.

In particular, the superiority of our approach is confirmed by a blind perceptual study. This work provides a new reference method for high-quality style transfer with unequaled multi-resolution depth. It also naturally extends the state of the art for UHR texture synthesis. The main drawback of our approach is that it remains computationally heavy, taking several minutes to produce an image. Nevertheless, it us up to the users to define their speed vs. quality trade-off, and we believe that our algorithm can be viewed as a new gold standard for practitioners wishing to achieve the highest style transfer image quality. Our public implementation (available at `https://github.com/bgalerne/scaling_painting_style_transfer`) will also allow future research on fast but approximate models to be compared with our method.

2. Related work

2.1. Style transfer by optimization

As recalled in the introduction, the seminal work of Gatys *et al.* formulated style transfer as an optimization problem minimizing the distances between Gram matrices of VGG features [GEB16]. Other global statistics have been proven effective for style transfer and texture synthesis such as deep correlations [SC17; GGL22], Bures metric [VDKC20], spatial mean of features [LZW16; DDD*21], feature histograms [RWB17], or even the full feature distributions [HVCB21]. Specific cost function corrections have also been proposed for photorealistic style transfer [LPSB17]. When dealing with HR images, a coarse-to-fine multiscale strategy has been proven efficient to capture the different levels of details present in style images [GEB*17; Sne17; GGL22]. Style transfer by optimization has also been extended for video style transfer [RDB16] and style transfer for neural fields [ZKB*22].

The original optimization approach [GEB16] was considered unfitted for UHR style transfer due to high memory requirements (limited to $1k^2$ px images [TFF*20]). This paper presents an algorithm that solves this very problem for UHR images.

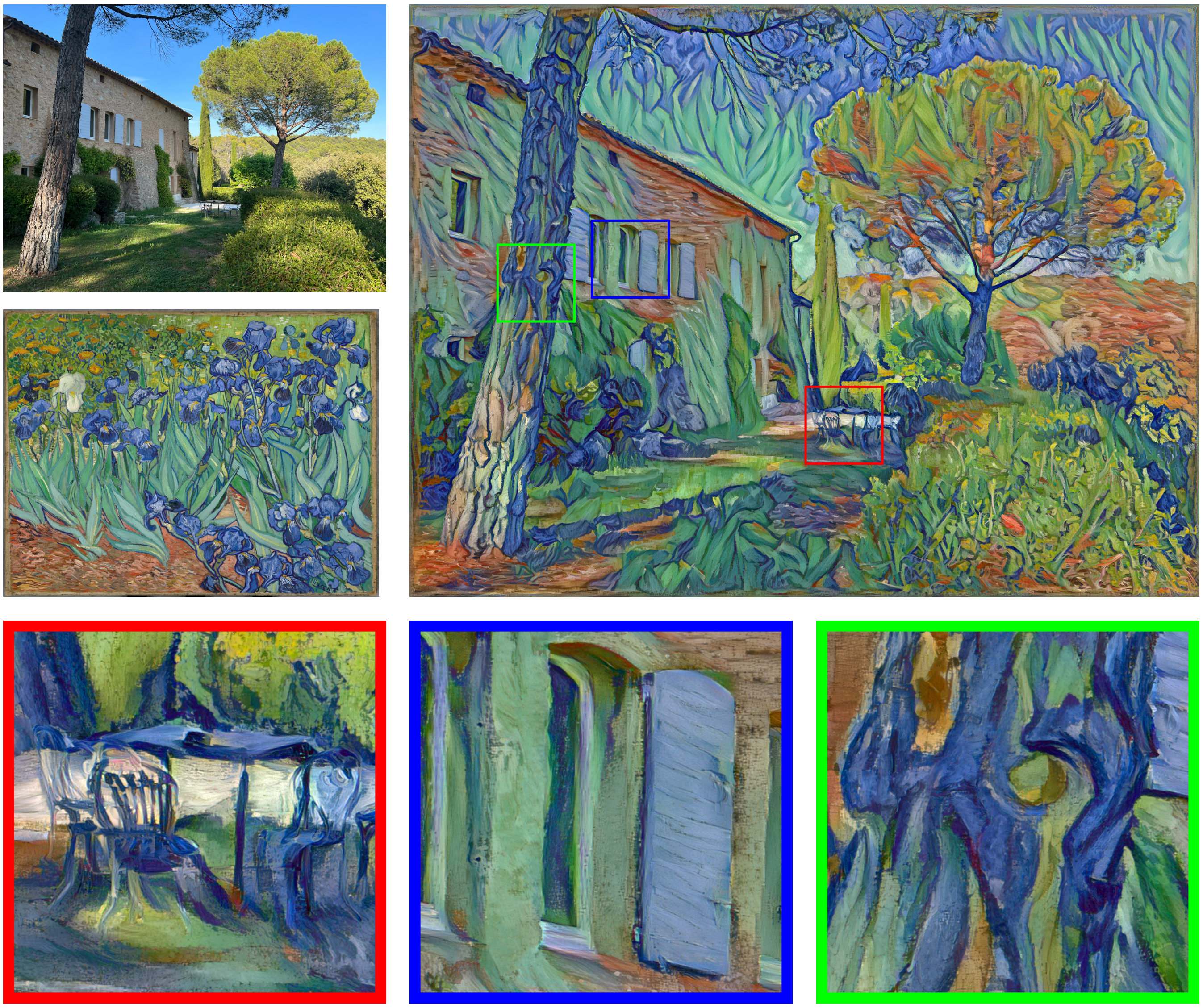

Figure 2: *UHR style transfer. Top row, content image (top-left, 6048×8064), style image (bottom left, 6048×7914), result (right, 6048×8064) (the three UHR images are downscaled ×4 for visualization). Bottom row: three zoomed in details of the result image (800^2, true resolution). Observe how very fine details such as the chairs look as if painted.*

2.2. Universal style transfer (UST)

Ulyanov *et al.* [ULVL16; UVL17] and Johnson *et al.* [JAF16] showed that feed-forward networks could be trained to approximately solve style transfer. Although these models produce a very fast style transfer, they require learning a new model for each style type, making them slower than the original optimization approach when training time is included.

Style limitation was addressed by training a VGG autoencoder that attempts to reverse VGG feature computations after normalizing them at the autoencoder bottleneck. Chen *et al.* [CS16] introduce the encoder-decoder framework with a style swap layer replacing content features with the closest style features on overlapping patches. Huang *et al.* [HB17] propose to use an Adaptive Instance Normalization (AdaIN) that adjusts the mean and variance of the content image features to match those of the style image. Li *et al.* [LFY*17] match the covariance matrices of the content image features to those of the style image by applying whitening and coloring transforms. These operations are performed layer by layer and involve specific reconstruction decoders at each step. Sheng *et al.* [SLSW18] use one encoder-decoder block combining the transformations of two previous work [LFY*17; CS16]. Park and Lee [PL19] introduce an attention-based transformation module to integrate the local style patterns according to the spatial distribution of the content image. Li *et al.* [LLKY19] train a symmetric encoder-decoder image reconstruction module and a transformation learning module. Chiu and Gurari [CG20] extend the UST approach of Li *et al.* [LFY*17] by embedding a new transformation that iteratively updates features in a cascade of four autoencoder modules. Despite the many improvements in fast UST strategies, we remark that: (a) they rely on matching VGG statistics as introduced in [GEB16] (b) they are limited in resolution due to GPU memory required for large size models.

2.3. UST for high-resolution images

Some methods attempt to reduce the size of the network in order to perform high resolution style transfer. ArtNet [ALH*20] is a channel-wise pruned version of GoogLeNet [SLJ*15]. Wang *et al.* [WLW*20] propose a collaborative distillation approach in order to compress the model by transferring the knowledge of a large network (VGG19) to a smaller one, hence reducing the number of convolutional filters used for UST [LFY*17; HB17]. Chen *et al.* [CWX*22] proposed an UHR style transfer framework where the content image is divided into patches and a patch-wise style transfer is performed from a zoomed out version of the style image of size 1024^2 px. Wang *et al.* [WZZ*23] recently proposed to avoid using pre-trained convolutional deep neural networks for inference and instead train three very lightweight models, a content encoder, a style encoder, and a decoder, resulting in a ultra-high resolution UST with very low inference time. However, as will be shown below, the UHR style transfer results generally suffer from visual artifacts and do not faithfully convey the complexity of the style painting at all scales.

Texler *et al.* [TFF*20] present a hybrid approach that combines neural networks and patch-based synthesis. They first perform NST between the low-resolution versions of the content and the style images, then refine the style details using patch-based transfer at a medium resolution followed by an upscaling. By design, this approach only consider a low-resolution version of the content image and suffers from a loss of details in comparison to our method (see supp. mat.). In addition to style transfer, other works have addressed HR image synthesis using generative adversarial networks such as HR texture synthesis by tiling features in the latent space of a generative adversarial network [FAW19] as well as HR image generation using a bi-level approach [LLC*21] based on StyleGAN2 [KLA*20].

3. Global optimization for neural style transfer

3.1. Single scale style transfer

Let us recall the algorithm of Gatys *et al.* [GEB16]. It solely relies on optimizing some VGG19 second-order statistics for changing the image style while maintaining some VGG19 features to preserve the content image's geometric features. Style is encoded through Gram matrices of several VGG19 layers, namely the set $\mathcal{L}_\mathrm{s} = \{\texttt{ReLU\_k\_1},\ k \in \{1,2,3,4,5\}\}$ while the content is encoded with a single feature layer $L_\mathrm{c} = \texttt{ReLU\_4\_2}$.

Given a content image u and a style image v, one optimizes the loss function

$$E_{\text{transfer}}(x;(u,v)) = E_{\text{content}}(x;u) + E_{\text{style}}(x;v) \tag{1}$$

where $E_{\text{content}}(x;u) = \lambda_\mathrm{c} \left\| V^{L_\mathrm{c}}(x) - V^{L_\mathrm{c}}(u) \right\|^2$, with $\lambda_\mathrm{c} > 0$, and

$$E_{\text{style}}(x;v) = \sum_{L\in\mathcal{L}_\mathrm{s}} E^L_{\text{style}}(x;v). \tag{2}$$

The style loss for a layer $L \in \mathcal{L}_\mathrm{s}$ is the Gram loss

$$E^L_{\text{style}}(x;v) = w_L \left\| G^L(x) - G^L(v) \right\|_\mathrm{F}^2, \quad w_L > 0, \tag{3}$$

where $\|\cdot\|_\mathrm{F}$ is the Frobenius norm, and, for an image w and a layer index L, $G^L(w)$ denotes the Gram matrix of the VGG19 features at layer L: if $V^L(w)$ is the feature response of w at layer L that has spatial size $n^L_\mathrm{h} \times n^L_\mathrm{w}$ and n^L_c channels, one first reshapes $V^L(w)$ as a matrix of size $n^L_\mathrm{p} \times n^L_\mathrm{c}$ with $n^L_\mathrm{p} = n^L_\mathrm{h} n^L_\mathrm{w}$ the number of feature pixels, its associated Gram matrix is the $n^L_\mathrm{c} \times n^L_\mathrm{c}$ matrix

$$G^L(w) = \frac{1}{n^L_\mathrm{p}} V^L(w)^\top V^L(w) = \frac{1}{n^L_\mathrm{p}} \sum_{k=0}^{n^L_\mathrm{p}} V^L(w)_k (V^L(w)_k)^\top, \tag{4}$$

where $V^L(w)_k \in \mathbb{R}^{n^L_\mathrm{c}}$ is the column vector corresponding to the k-th line of $V^L(w)$. $E^L_{\text{style}}(x;v)$ is a fourth-degree polynomial and non convex with respect to (wrt) the VGG features $V^L(x)$. Gatys *et al.* [GEB15] propose to use the L-BFGS algorithm [Noc80] to minimize this loss, after initializing x with the content image u. L-BFGS is an iterative quasi-Newton procedure that approximates the inverse of the Hessian using a fixed size history of the gradient vectors computed during the last iterations.

3.2. Gram loss correction

Previous works [SC17; RWB17; HVCB21] have shown that optimizing the Gram loss alone may introduce some loss of contrast artifacts. A proposed explanation is that Gram matrices encompass information regarding both the mean values and correlation of features. While is has been shown that reproducing the full histogram of the features [RWB17; HVCB21] permits to avoid this artefact, we found that simply correcting for the mean and standard deviation (std) of each feature produced visually satisfying results and is computationally simpler.

Given some (reshaped) features $V \in \mathbb{R}^{n_\mathrm{p}\times n_\mathrm{c}}$, define $\mathrm{mean}(V)$ and $\mathrm{std}(V) \in \mathbb{R}^{n_\mathrm{c}}$ as the spatial mean and standard deviation vectors of each feature channel. Throughout the paper, the Gram loss $w_L \left\| G^L(u) - G^L(v) \right\|_\mathrm{F}^2$ of Eq. (3) is replaced by the following augmented style loss

$$\begin{aligned} \tilde{E}^L_{\text{style}}(x;v) = & w_L \left\| G^L(u) - G^L(v) \right\|_\mathrm{F}^2 \\ & + w'_L \| \mathrm{mean}(V^L(x)) - \mathrm{mean}(V^L(v)) \|^2 \\ & + w''_L \| \mathrm{std}(V^L(x)) - \mathrm{std}(V^L(v)) \|^2 \end{aligned} \tag{5}$$

for a better reproduction of the feature distribution. Note that using the "mean plus std loss" alone was proposed in [LWLH17] as an alternate loss for NST (see also [HB17]). The values of all the weights $\lambda_\mathrm{c}, w_L, w'_L, w''_L, L \in \mathcal{L}_\mathrm{s}$, have been fixed for all images (see the provided source code for the exact values).

Limiting our style loss $\tilde{E}^L_{\text{style}}(x;v)$ to second-order statistics is capital for our localized algorithm described in Section 4. Indeed, using more involved techniques such as slice Wasserstein distance minimization [HVCB21] is not feasible for UHR images due to prohibitive memory requirement. The visual improvement when replacing E^L_{style} by $\tilde{E}^L_{\text{style}}$ is illustrated in the supp. mat.

3.3. Limited resolution

Unfortunately, applying this Gatys *et al.* algorithm off-the-shelf with UHR images is not possible in practice for images of size larger than 4000 px, even with a high-end GPU. The main limitation comes from the fact that differentiating the loss $E_{\text{transfer}}(x;(u,v))$ wrt x requires fitting into memory x and all its intermediate VGG19 features. While this requires a moderate 2.61 GB for a 1024^2 px image, it requires 10.2 GB for a 2048^2 while scaling up to 4096^2 is not feasible with a 40 GB GPU. In the next section we describe a practical solution to overcome this limitation.

4. Localized style transfer loss gradient

As mentioned in the introduction, our main contribution is to emulate the computation of

$$\nabla_x E_{\text{transfer}}(x;(u,v)) \tag{6}$$

even for images larger than 4000^2 px for which evaluation and automatic differentiation of the loss is not feasible due to large memory requirements.

We first discuss how one can compute neural features in a localized way and straightforwardly compute the style transfer loss using a spatial partition of the image. Then, we demonstrate that this approach allows for the exact computation of the loss gradient using a two-pass procedure.

4.1. Localized computation of neural features

First suppose one wants to compute the feature maps $V^L(x)$, $L \in \mathcal{L}_s \cup \{L_c\}$, of an UHR image x. The natural idea developed here is to compute the feature maps piece by piece, by partitioning the input image x into small images of size 512^2, that we will call blocks. This approach will work up to boundary issues. Indeed, to compute exactly the feature maps of x one needs the complete receptive field centered at the pixel of interest. Hence, each block of the partition must be extracted with a margin area, except on the sides that are actual borders for the image x. In all our experiments we use a margin of width 256 px in the image domain.

This localized way to compute features allows one to compute global feature statistics such as Gram matrices and means and stds vectors. Indeed, these statistics are all spatial averages that can be aggregated block by block by adding sequentially the contribution of each block. Hence, this easy to implement procedure allows one to compute the value of the loss $E_{\text{transfer}}(x;(u,v))$ (see Equation (1)). However, in contrast with standard practice, it is *not* possible to automatically differentiate this loss wrt x, because the computation graph linking back to x has been lost.

4.2. Localized gradient given global statistics

A close inspection of the different style losses wrt the neural features shows that they all have the same form: For each style layer $L \in \mathcal{L}_s$, the gradient of the layer style loss $\tilde{E}^L_{\text{style}}(x;v)$ wrt the layer feature $V^L(x)_k \in \mathbb{R}^{n^L_c}$ at some pixel location k only depends on the local value $V^L(x)_k$ and on some difference between the global statistics (Gram matrix, spatial mean, std) of $V^L(x)$ and the corresponding ones from the style layer $V^L(v)$.

Proposition 4.1 (Locality of style loss gradient) Given the layer global statistics values, the gradient of the layer style loss $\tilde{E}^L_{\text{style}}(x;v)$ wrt the layer feature $V^L(x) \in \mathbb{R}^{n^L_c}$ is local: The gradient value at location k only depends on the feature $V^L(x)_k$ at the same location k.

Proof Recall from Equation (5) that $\tilde{E}^L_{\text{style}}(x;v)$ is a linear combination of the Gram, mean, and std losses. As shown in Table 1, given the global statistics, each of these losses satisfies the local property. $\square$

Exploiting this locality of the gradient, it is also possible to *exactly* compute the gradient vector $\nabla_x E_{\text{transfer}}(x;(u,v))$ block by block using a two-pass procedure: The first pass is used to compute the global VGG19 statistics of each style layer and the second pass is used to locally backpropagate the gradient wrt the local neural features. The whole procedure is described by Algorithm 1 and illustrated by Figure 3. As illustrated by Figure 4, Algorithm 1 enables to exactly compute the global gradient of the loss in a localized way. The used block margin of size 256 is necessary to avoid visual discontinuities at block boundaries (see Figure 4).

5. Multiscale high-resolution painting style transfer

5.1. Coarse-to-fine style transfer

Thanks to Algorithm 1, we can apply style transfer to unprecedented scales. However, applying a direct style transfer to UHR images generally does not produce the desired effects due to the fixed size of VGG19 receptive fields. For images larger than 500^2 px, visually richer results are obtained by adopting a multiscale approach [GEB*17] corresponding to the standard coarse-to-fine texture synthesis [WL00] that we recall in Algorithm 2.

Our two step localized computation approach allows to apply style transfer through up to 6 scales (e.g., from 512^2 px to 16384^2 px). Except for the first step, all subsequent style transfers are well-initialized, allowing for a faster optimization [GEB*17]. For our baseline implementation, we use L-BFGS with 600 iterations for the first scale and 300 iterations for the subsequent scales. Due to the large memory needed to store UHR images, the L-BFGS history is limited to the 10 last gradients for all scales except the first one that uses the standard history size of 100.

Finally, in order to avoid GPU memory saturation, for very large images we perform the L-BFGS update procedure and gradient history storing on the CPU for the last scale. This allows to increase the maximal number of pixels by 190% (+70% in square image side), as reported in the left column of Table 2. In particular this allows to apply style transfer on images with the unprecedented size of 20k^2 using a GPU with 80 GB of memory.

The coarse-to-fine procedure is revealed to be essential to convey the visual complexity of UHR digital photograph of a painting: the first scale encompasses color and large strokes while subsequent

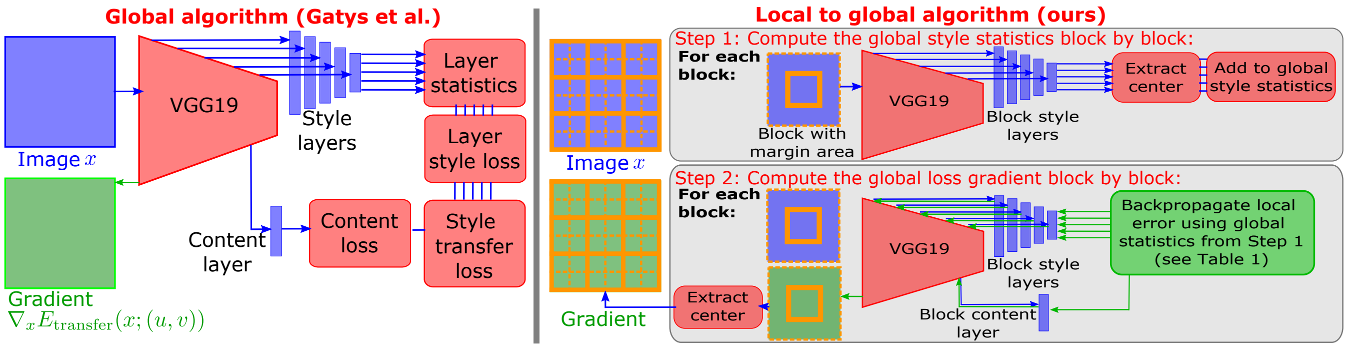
Global algorithm (Gatys et al.)
Image x
VGG19
Style layers
Layer statistics
Layer style loss
Content layer
Content loss
Style transfer loss
Gradient $\nabla_x E_{\text{transfer}}(x;(u,v))$
Local to global algorithm (ours)
Step 1: Compute the global style statistics block by block:
For each block:
Block with margin area
VGG19
Block style layers
Extract center
Add to global style statistics
Image x
Step 2: Compute the global loss gradient block by block:
For each block:
VGG19
Block style layers
Backpropagate local error using global statistics from Step 1 (see Table 1)
Extract center
Block content layer
Gradient

Figure 3: *Algorithm overview: Our localized algorithm (right part) allows to compute the global style transfer loss and its gradient wrt x for images that are too large for the original algorithm of [GEB16] (left part). See Algorithm 1 for the fully detailed procedure.*

Global statistics	Feature loss Expression	Gradient wrt the feature
Raw features V	MSE: $E(V) = \|V - V_{\text{ref}}\|^2$	$\nabla_V E(V) = 2(V - V_{\text{ref}})$
Gram matrix: $G = \frac{1}{n_{\text{p}}} V^\top V$	Gram loss: $E(V) = \|G - G_{\text{ref}}\|_{\text{F}}^2$	$\nabla_V E(V) = \frac{4}{n_{\text{p}}} V(G - G_{\text{ref}})$
Feature mean: $\text{mean}(V)$	Mean loss: $E(V) = \|\text{mean}(V) - \mu_{\text{ref}}\|^2$	$(\nabla_V E(V))_k = \frac{2}{n_{\text{p}}}(\text{mean}(V) - \mu_{\text{ref}})$
Feature std: $\text{std}(V)$	Std loss: $E(V) = \|\text{std}(V) - \sigma_{\text{ref}}\|^2$	$\nabla_V E(V)_{k,j} = \frac{2}{n_{\text{p}}}(V_{k,j} - (\text{mean}(V))_j)\frac{(\text{std}(V))_j - \sigma_{\text{ref},j}}{(\text{std}(V))_j}$

Table 1: *Expression of the feature loss gradient wrt a generic feature V having n_{p} pixels and n_{c} channels (matrix size $n_{\text{p}} \times n_{\text{c}}$).*

GPU (VRAM)		Computation time (in min.)			
w/ max. res. GPU / GPU+CPU		2k	4k	8k	16k
RTX 2080 (11GB)	SPST	12.8	70.3*	-	-
max. res. 3k / 5k*	SPST-fast	4.3	13.1*	-	-
A100 (40GB)	SPST	4.0	20.6	96.6	-
max. res. 8k / 14k*	SPST-fast	1.7	4.1	15.2	-
A100 (80GB)	SPST	3.8	19.0	89.7	406*
max. res. 12k / 20k*	SPST-fast	1.5	3.9	14.4	61.4*

Table 2: *Resolution and computation time of SPST and SPST-fast depending on GPU hardware. Below each hardware name, we give the maximum resolution achievable with full computation on the GPU and the maximum resolution achievable when using the CPU for L-BGFS steps for the last scale (denoted by *). For the computation times, * indicates that L-BFGS optimization had to be moved to the CPU to avoid GPU memory saturation. All images are square.*

scales refine the stroke details up to the painting texture, bristle brushes, fine painting cracks and canvas texture, as illustrated in Figure 1. Surprisingly, fast methods for universal style transfer are not based on a coarse-to-fine approach, which is probably the main reason for their lack of fidelity to fine details (see Section 7.1).

5.2. Accelerated multiscale style transfer

The main drawback of our baseline approach is the computational cost. Indeed, the complexity is linear in the number of pixels, making each upscaling step four times longer than the previous one. Nevertheless, we experimentally observed that the style transfer is remarkably stable from one step to the next, as can be observed in the top row of Figure 5. To the best of our knowledge, this property has never been reported, probably because style transfer involving several scales was not reachable without our localized algorithm for gradient computation.

The role of the last steps is to refine local texture in accordance to the style image at the current resolution. To allow for a faster alternative, we found that these last steps can be alleviated by reducing the number of iterations. We thus propose an alternative procedure, called *SPST-fast* in what follows, that reduces the number of iterations by a factor 3 from one scale to the other, while ensuring a minimal number of 30 iterations, e.g., for 4 scales one uses $(n_{\text{it}}^s)_{1\leq x\leq 4} = (600, 200, 66, 30)$ instead of $(n_{\text{it}}^s)_{1\leq x\leq 4} = (600, 300, 300, 300)$ for the baseline implementation. Computation times for both SPST and SPST-fast are reported in Table 2 for three different GPU hardwares. They show that SPST-fast is about five times faster than SPST. Note that our algorithm allows for multiscale style transfer of UHR images up to 20k^2 px. Even on a moderate GPU with 11 GB of memory, our algorithm can deal with images of size 5k^2 px, while the original implementation of [GEB16] does not run on a 40GB GPU for an image of size 4k^2 px. Our source code is available at `https://github.com/bgalerne/scaling_painting_style_transfer`.

As shown in Figure 5, SPST-fast produces visually satisfying results but with small texture details that are slightly less aligned with the UHR content image compared to the SPST baseline approach.

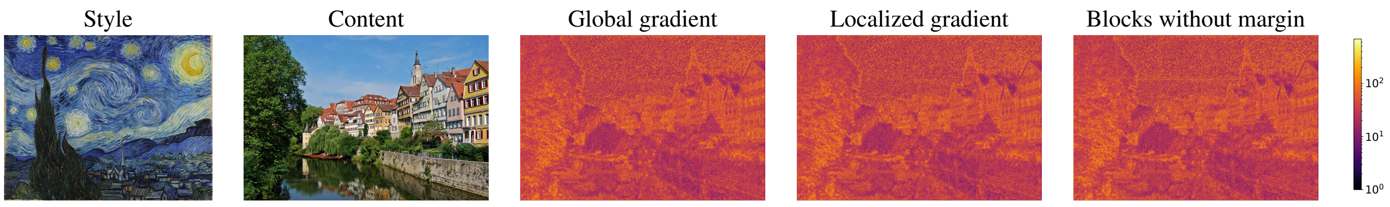

Figure 4: *Localized gradient computation: From left to right: Style image (size 1953×2466), content image (size 1953×2900), norm of the RGB gradient at each pixel computed with three different approaches: reference global gradient [GEB16], localized gradient using Algorithm 1 (blocks of size 512×512, block margin is 256), and localized gradient with block margin set to zero. Algorithm 1 allows for the exact computation of the gradient up to numerical errors (relative error is 1.03e-2). Using a block margin of size zero instead of 256 produces a gradient with visible seams at block boundaries (relative error is 5.23e-1).*

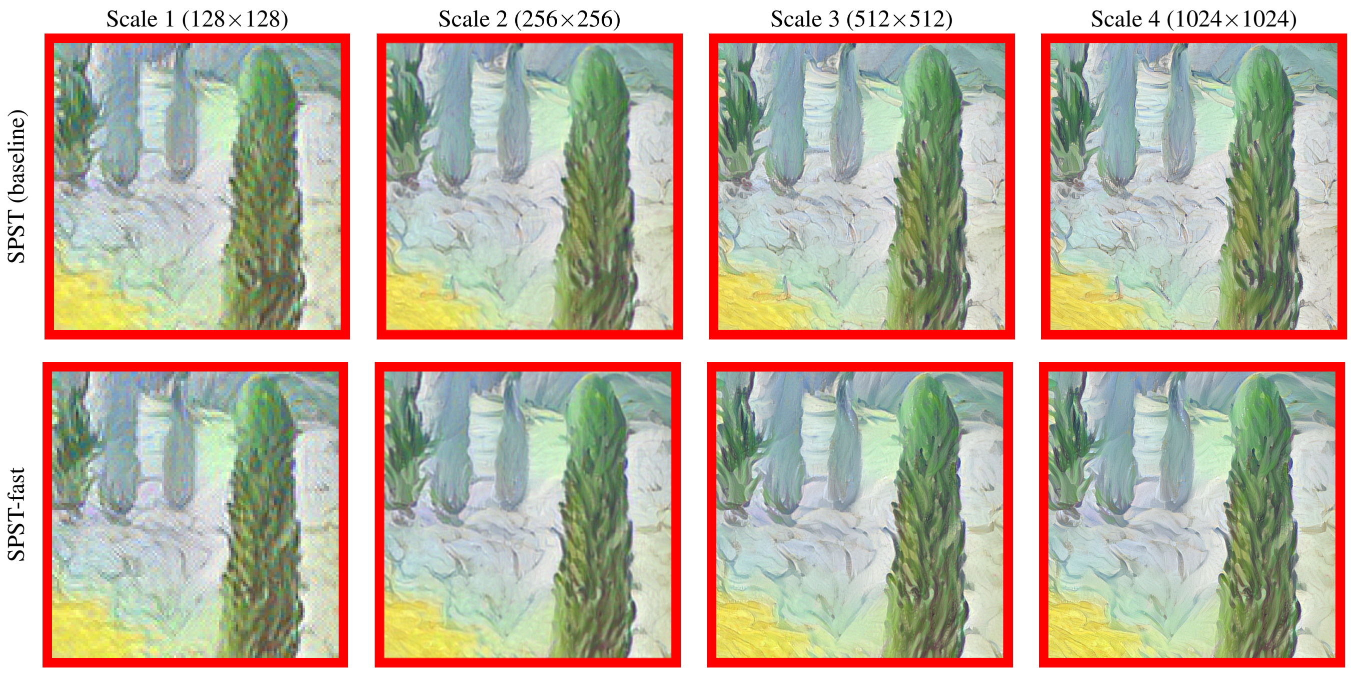

Figure 5: *UHR multiscale style transfer: Stability of upscaling and SPST-fast. The top row shows intermediary steps for the experiment of Figure 1 displaying details of size* 128^2 *to* 1024^2*. While the transfer is globally stable from one scale to the next, each upscaling enables the addition of fine pictorial details which give an authentic painting aspect to the final output image. Bottom row: Same details for the output of the SPST-fast alternative that uses less and less L-BFGS iterations after each upscaling. Computation times for the full scale image outputs of size 6048×8064 are 74 minutes for SPST and 13 minutes for SPST-fast (×5.6 speed up). Observe that both methods produce very close results but that the very fine details of the SPST-fast output are slightly less complex.*

6. Numerical Results

6.1. Ultra-high resolution style transfer

An example of UHR style transfer is displayed in Figure 2 with several highlighted details. Figure 1 illustrates intermediary steps of our high resolution multiscale algorithm. The result for the first scale (third column) corresponds to the ones of the original paper [GEB16] (except for our slightly modified style loss) and suffers from poor image resolution and grid artifacts. As already discussed with Figure 5, while progressing to the last scale, the texture of the painting gets refined and stroke details gain a natural aspect. This process is remarkably stable; the successive global style transfers results remain consistent with the one of the first scale.

6.2. Ultra-high resolution texture synthesis

Although we focus our discussion on style transfer, our approach also allows for UHR texture synthesis. Following the original paper on texture synthesis [GEB15], given a texture exemplar v, texture synthesis is performed by minimizing $E_{\text{style}}(x;v)$ (2), starting from a random white noise image x_0. From a practical point of view, it consists in minimizing the style transfer loss with the three follow-

Algorithm 1 Localized computation of the style transfer loss and its gradient wrt x

```
Input: Current image $x$, content image layer $V^{L_c}(u)$, and list of feature statistics of $v$ $\{(G^L(v), \text{mean}(V^L(v)), \text{std}(V^L(v))),\ L \in \mathcal{L}_s\}$ (computed block by block)
Output: $E_{\text{transfer}}(x;(u,v))$ and $\nabla_x E_{\text{transfer}}(x;(u,v))$
  Step 1: Compute the global style statistics of $x$ block by block:
  for each block in the partition of $x$ do
      Extract the block $b$ with margin and compute VGG($b$) without computation graph
      For each style layer $L \in \mathcal{L}_s$: Extract the features of the block by properly removing the margin and add their contribution to $G^L(x)$ and mean$(V^L(x))$.
  end for
  For each style layer $L \in \mathcal{L}_s$: Compute
  $\text{std}(V^L(x)) = (\text{diag}(G^L(x)) - \text{mean}(V^L(x))^2)^{\frac{1}{2}}$.
  Step 2: Compute the transfer loss and its gradient wrt $x$ block by block:
  Initialize the loss and its gradient: $E_{\text{transfer}}(x;(u,v)) \leftarrow \tilde{E}_{\text{style}}(x;v)$; $\nabla_x E_{\text{transfer}}(x;(u,v)) \leftarrow 0$
  for each block in the partition of $x$ do
      Extract the block $b$ with margin and compute VGG($b$) with computation graph
      For each style layer $L \in \mathcal{L}_s$: Compute the gradient of the style loss wrt the local features using the global statistics of $x$ from Step 1 and the style statistics of $v$ as reference (Table 1)
      For the content layer $L_c$, add the contribution of $V^{L_c}(b)$ to the loss $E_{\text{transfer}}(x;(u,v))$ and compute the gradient of the content loss wrt the local features (first row of Table 1)
      Use automatic differentiation to backpropagate all the feature gradients to the level of the input block image $b$.
      Populate the corresponding block of $\nabla_x E_{\text{transfer}}(x;(u,v))$ with the inner part of the gradient obtained by backpropagation.
  end for
```

Algorithm 2 Multiscale style transfer

```
Input: Content image $u$, a style image $v$, number of scales $n_{\text{scales}}$
Output: Style transfered image $x$
  for scale $s = 1$ to $n_{\text{scales}}$ do
      Downscale $u$ and $v$ by a factor $2^{n_{\text{scales}}-s}$ to obtain the low-resolution couple $(u^{\downarrow}, v^{\downarrow})$
      Initialization: If $s = 1$ let $x = u^{\downarrow}$, otherwise upscale current $x$ by a factor 2
      Style transfer at current scale:
      $x^{\downarrow} \leftarrow$ StyleTransfer$((u^{\downarrow}, v^{\downarrow}), x^{\downarrow})$ using $n_{\text{it}}^s$ iterations of L-BFGS with gradient computed with Algorithm 1
  end for
```

ing differences: a) The style image is replaced by the texture image. b) There is no content image and no content loss (set $\lambda_c = 0$). c) The image x is initialized as a random white noise x_0. We perform texture synthesis following the same multiscale approach and using the augmented style loss $\tilde{E}^L_{\text{style}}(x;v)$ defined in Equation (5).

Our experiments show that for texture synthesis, one should use a number of scales as high as possible, that is, the multiscale process starts with images of moderate size (about 200 pixels). To illustrate this point we show two different UHR texture synthesis in Figures 6 (six additional results are displayed in the supp. mat.). For each example, the synthesis using three scales (same setting as for style transfer) and five scales is shown. Starting with a first scale with small size is critical for a satisfying synthesis quality. Indeed, using only three scales yields textures that are spatially homogeneous due to the white noise initialization.

Let us recall that our approach enables to reach up to 20k^2 px (see Table 2 for maximal resolutions), which pushes by far the maximal resolution for neural texture synthesis. Indeed, to the best of our knowledge the highest resolution reported in the neural texture synthesis literature was limited to 2048^2 px [GGL22] for the multiscale version of Gatys *et al.* algorithm [GEB15].

7. Comparison with very fast alternatives

7.1. Visual comparison

We compare our method with two fast alternatives for UHR style transfer, namely collaborative distillation (CD) [WLW*20] and URST [CWX*22] (based on [LFY*17]) using their official implementations. To improve readability of Figure 8, results for SPST-fast, which are really close to the ones of SPST but have slightly less details, are only reproduced in supp. mat.

As already discussed in Section 2, URST decreases the resolution of the style image to 1024^2 px, so the style transfer is not performed at the proper scale and fine details cannot be transferred (e.g., the algorithm is not aware of the brushstroke style). As in UST methods, CD does not take into account details at different scales but simply proposes to reduce the number of filters in the auto-encoder network through collaborative distillation, to process larger images. Unsurprisingly, one observes in Figure 8 that our method is the only one capable of conveying the aspect of the painting strokes to the content image. CD suffers from halos around objects (e.g., tress in the first example), saturated color, and high-frequency artifacts (see fourth column of Figure 8). URST presents visible patch boundaries, a detail frequency mismatch due to improper scaling, loss of structure (e.g., buildings in the second example) and sometimes critical shrinking of the color palette (see fifth column of Figure 8).

All in all, even though CD and URST produce UHR images, one can argue that the effective resolution of the output does not match their size due to the many visual artifacts. In comparison, our iterative SPST algorithm produces images for which every image part is in accordance with UHR painting style, up to the pixel level.

Finally, let us observe that the style transfer results are in general better when the geometric content of the style image and the content image are close, regardless of the method (see Figure 7

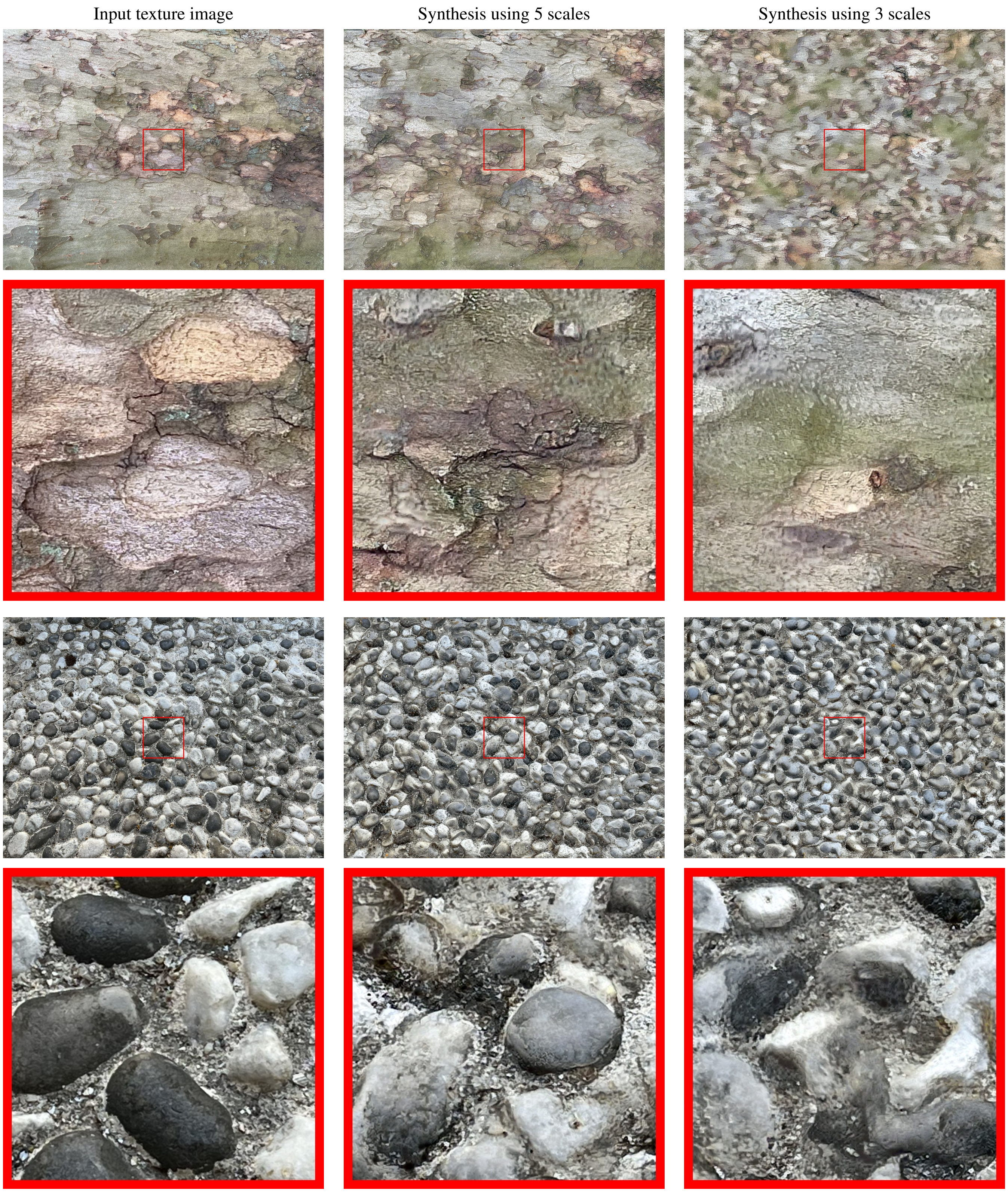

Figure 6: *UHR texture synthesis (same as Figure 6): From left to right: Input texture image, synthesis using 5 scales, synthesis using 3 scales. Image have size (3024×4032) (downscaled by a factor 4 for inclusion in the .pdf) and true resolution details have size (512×512).*

Method	PSNR ↑	SSIM ↑	LPIPS ↓	Gram ↓	Time ↓
SPST	<u>24.6</u>	**0.454**	**0.352**	**1.99e5**	25.1
SPST-fast	**24.6**	<u>0.438</u>	<u>0.446</u>	<u>4.08e5</u>	4.96
CD	21.8	0.413	0.500	4.28e7	<u>0.373</u>
URST	19.0	0.413	0.546	6.77e7	**0.232**

Table 3: *Quantitative evaluation of* identity test *for UHR style transfer. Results include PSNR, SSIM [WBSS04], LPIPS [ZIE*18], the Gram (*style distance*) metrics, and computation time in minutes for our results (SPST), its faster alternative (SPST-fast), CD [WLW*20] and URST [CWX*22]. All metrics are averages using 79 HR paintings used as both content and style. Best results are in bold, second best underlined. Our iterative procedures SPST and SPST-fast are the best for all the image fidelity metrics but are respectively 100 and 20 times slower.*

and supp. mat. for additional examples). Gatys *et al.* [GEB*17] show that one can mitigate these failure cases by adding more control (e.g., segmentation, color transfer as preprocessing, careful rescaling of the style). All these solutions can be straightforwardly adapted to work on UHR images.

7.2. Identity test for style transfer quality assessment

Style transfer is an ill-posed problem by nature. We introduce here an *identity test* to evaluate if a method is able to reproduce a painting when using the same image for both content and style. Two examples of this sanity check test are shown in Figure 9. We observe that our iterative algorithm is slightly less sharp than the original style image, yet high-resolution details from the paint texture are faithfully conveyed. In comparison, the results of CD [WLW*20] suffer from color deviation and frequency artifacts while URST [CWX*22] applies a style transfer that is too homogeneous and present color and scale issues as already discussed. Again corresponding results for Figure 9 for SPST-fast are only reproduced in supp. mat. for readability.

Some previous works introduced a *style distance* [WLW*20] that corresponds to the Gram loss for some VGG19 layers, showing again that fast approximate methods try to reproduce the algorithm of Gatys *et al.* which we extend to UHR images. Since we explicitly minimize this quantity, it is not fair to only consider this criterion for a quantitative evaluation. For this reason, we also calculated PSNR, SSIM [WBSS04] and LPIPS [ZIE*18] metrics on a set of 79 paint styles (see supp. mat.) to quantitatively evaluate our results. We further report the “Gram” metric, that is, the style loss of Equation (2) using the original Gram loss of Equation (3), computed on UHR results using our localized approach. The average scores reported in Table 3 confirm the good qualitative behavior discussed earlier: SPST and SPST-fast are by far the best for all the scores. However, SPST and SPST-fast are respectively 100 and 20 times slower than the fastest method.

	Voting results (%)		
	Id global	Id detail	Style transfer
CD	<u>6.56</u>	<u>22.95</u>	<u>4.92</u>
URST	0.33	2.29	4.26
SPST-fast	**93.11**	**74.75**	**90.82**

Table 4: *This perceptual study results shows the percentage of times each method was selected out of the 305 comparisons for each experiment. Best results are in bold, second best underlined.*

7.3. Perceptual study

To further compare our results, we performed a perceptual study comparing the fast version of our algorithm (SPST-fast) to CD [WLW*20] and URST [CWX*22].

The perceptual study consisted of several evaluation instances, each of which compared four images: the style used for the transfer and the results of the three methods (SPSt-fast, URST, and CD), which were displayed at random positions for each evaluation instance. Each participant was asked to select the result closest to the style of the style image among the three displayed results.

Participants were presented three types of experiments, each of which had five instances to evaluate, thus yielding a total of 15 instances to evaluate per candidate. The results were saved only if the participant conducted the whole test. The first two experiments aim to compare the results of our identity test. In one case, the overall performance of the methods is evaluated by displaying the complete results at a resolution of 1280×720, and in the other case, the performance of the methods on fine details is evaluated by displaying a close-up of the results at a size of 512×512. For the identity test, 79 painting styles were used and each participant was shown five random instances for the global evaluation and another five for the detail evaluation. The third experiment aims to compare the results of the three methods when transferring a painting style image to a generic content image. Only the overall performance of the methods is compared displaying the whole results at a resolution of 1280×720. In this case, 13 pairs of style/content images were used, and five instances were randomly shown to each participant.

A total of 61 participants took the test, yielding a total of 305 evaluations for each type of experiment. All invited participants were image processing experts in academy and industry. The results of the study are shown in Table 4. They confirm that our approach, both for the identity test (global and close-up) and the transfer of a painting style to any image, is by far superior to CD and URST in terms of visual quality: Our method is considered better more than 90% of the times as the one that better reproduces the style of the painting (Table 4 third column).

8. Conclusion

This work presented the SPST algorithm, a provably correct extension of the Gatys *et al.* style transfer algorithm to UHR images. Regarding visual quality, our algorithm outperforms competing UHR methods by conveying a true painting feel thanks to faithful HR details such as strokes, paint cracks, and canvas texture. This is

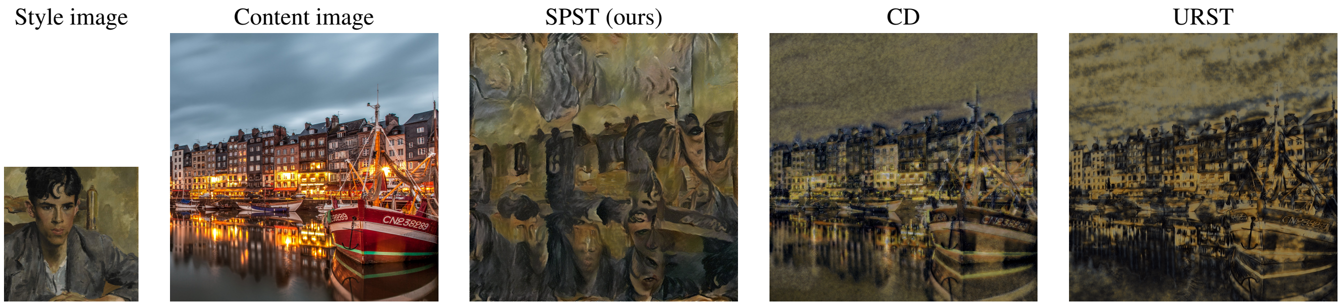

Figure 7: *Example of content-style mismatch. From left to right: style (2048×2048), content (4096×4096), our results (SPST), CD [WLW*20] and URST [CWX*22]. We used four scale for our results. The content-style mismatch results in the loss of details in the buildings and does not preserve the homogeneity of the sky.*

clearly supported by our perceptual study and our proposed quantitative *identity test*. SPST also allows for the synthesis by example of high-quality UHR textures. While the baseline SPST method can become prohibitively slow, even though its complexity scales linearly with image size, we proposed a faster alternative SPST-fast that limits computations as the scale grows by exploiting the stability of multiscale style transfer.

As we have demonstrated, very fast methods do not reach a satisfying quality. They fail our proposed identity test due to the presence of many artifacts, and our results are considered more faithful to the style image by a vast majority of users. This work also leads to conclude that very fast high-quality style transfer remains an open problem and that our results provide a new standard to assess the overall quality of such algorithms.

This work opens the way for several future research directions, from allowing local control for UHR style transfer [GEB*17] to training fast CNN-based models to reproduce our results. Another promising direction is to extend our framework to video or radiance fields style transfer for which reaching ultra-high resolution would be beneficial.

Acknowledgements: B. Galerne and L. Raad acknowledge the support of the project MISTIC (ANR-19-CE40-005).

References

[ALH*20] An, Jie, Li, Tao, Huang, Haozhi, et al. "Real-time universal style transfer on high-resolution images via zero-channel pruning". *arXiv preprint arXiv:2006.09029* (2020) 2, 4.

[APH*14] Aubry, Mathieu, Paris, Sylvain, Hasinoff, Samuel W., et al. "Fast Local Laplacian Filters: Theory and Applications". *ACM Trans. Graph.* 33.5 (Sept. 2014). ISSN: 0730-0301. DOI: 10.1145/2629645 2.

[CG20] Chiu, Tai-Yin and Gurari, Danna. "Iterative Feature Transformation for Fast and Versatile Universal Style Transfer". *European Conference on Computer Vision*. Springer. 2020, 169–184 2, 4.

[CS16] Chen, Tian Qi and Schmidt, Mark. "Fast patch-based style transfer of arbitrary style". *arXiv preprint arXiv:1612.04337* (2016) 2, 3.

[CWX*22] Chen, Zhe, Wang, Wenhai, Xie, Enze, et al. "Towards Ultra-Resolution Neural Style Transfer via Thumbnail Instance Normalization". *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 1. June 2022, 393–400. DOI: 10.1609/aaai.v36i1.19916 2, 4, 8, 10–13.

[DDD*21] De Bortoli, Valentin, Desolneux, Agnès, Durmus, Alain, et al. "Maximum Entropy Methods for Texture Synthesis: Theory and Practice". *SIAM Journal on Mathematics of Data Science* 3.1 (2021), 52–82. DOI: 10.1137/19M1307731 2.

[FAW19] Frühstück, Anna, Alhashim, Ibraheem, and Wonka, Peter. "Tilegan: synthesis of large-scale non-homogeneous textures". *ACM Transactions on graphics (TOG)* 38.4 (2019), 1–11 4.

[GEB*17] Gatys, Leon A., Ecker, Alexander S., Bethge, Matthias, et al. "Controlling Perceptual Factors in Neural Style Transfer". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017 2, 5, 10, 11.

[GEB15] Gatys, L., Ecker, A. S., and Bethge, M. "Texture Synthesis Using Convolutional Neural Networks". *Advances in Neural Information Processing Systems 28*. 2015, 262–270. URL: http://papers.nips.cc/paper/5633-texture-synthesis-using-convolutional-neural-networks.pdf 2, 4, 7, 8.

[GEB16] Gatys, L. A., Ecker, A. S., and Bethge, M. "Image Style Transfer Using Convolutional Neural Networks". *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016, 2414–2423. DOI: 10.1109/CVPR.2016.265 2, 4, 6, 7.

[GGL22] Gonthier, Nicolas, Gousseau, Yann, and Ladjal, Saïd. "High-Resolution Neural Texture Synthesis with Long-Range Constraints". *Journal of Mathematical Imaging and Vision* 64.5 (2022), 478–492 2, 8.

[HB17] Huang, Xun and Belongie, Serge. "Arbitrary Style Transfer in Real-Time With Adaptive Instance Normalization". *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017 2–4.

[HJO*01] Hertzmann, Aaron, Jacobs, Charles E., Oliver, Nuria, et al. "Image Analogies". *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '01. New York, NY, USA: Association for Computing Machinery, 2001, 327–340. ISBN: 158113374X. DOI: 10.1145/383259.383295 2.

[HVCB21] Heitz, Eric, Vanhoey, Kenneth, Chambon, Thomas, and Belcour, Laurent. "A Sliced Wasserstein Loss for Neural Texture Synthesis". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, 9412–9420 2, 4, 5.

[JAF16] Johnson, Justin, Alahi, Alexandre, and Fei-Fei, Li. "Perceptual losses for real-time style transfer and super-resolution". *European Conference on Computer Vision*. 2016, 694–711. DOI: 10.1007/978-3-319-46475-6_43 2, 3.

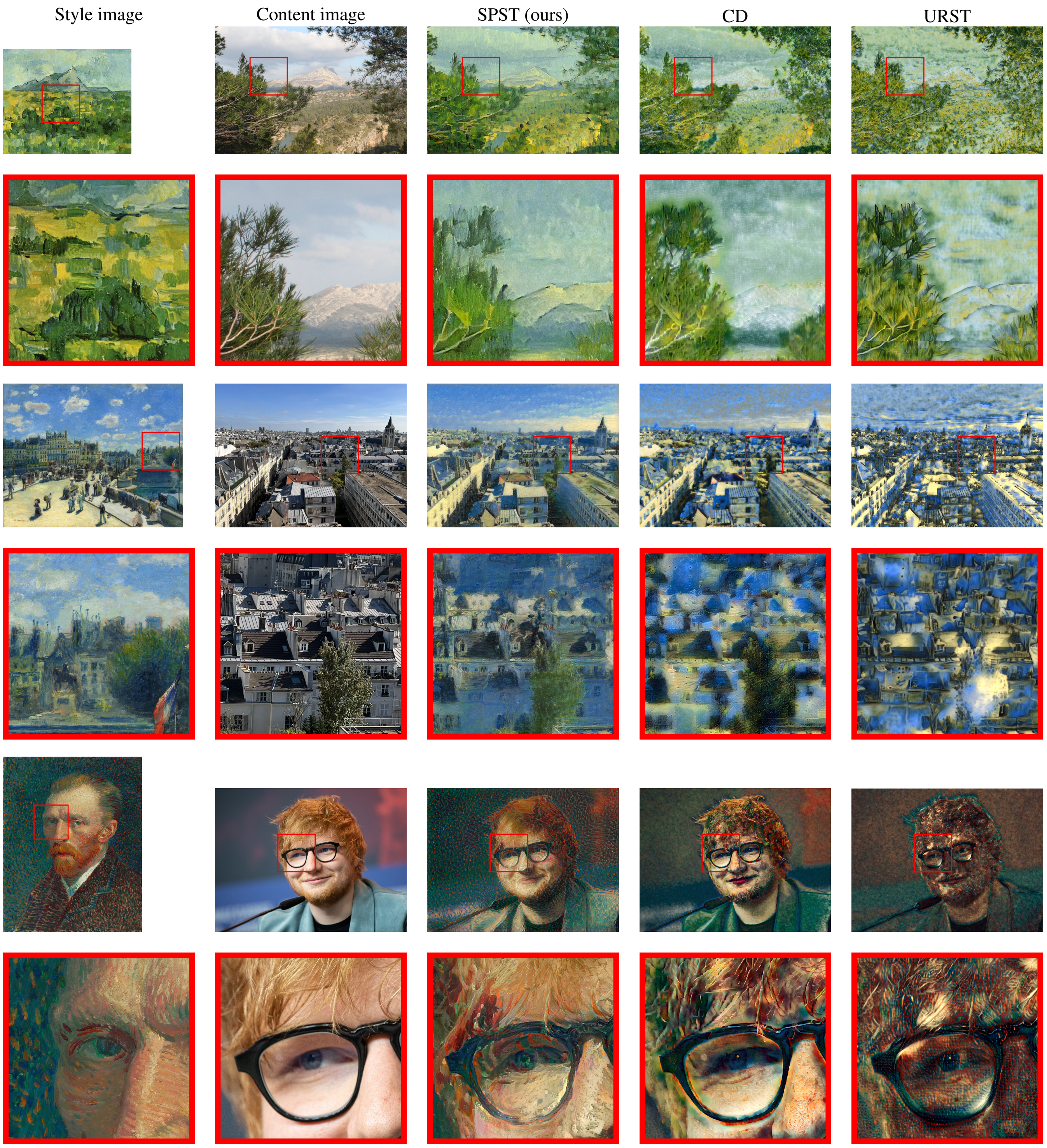

Figure 8: *Comparison of UHR style transfers. For each example, top row, left to right: style, content, our result (SPST), CD [WLW*20], URST [CWX*22]. Bottom row: zoom in of the corresponding top row. First row: content* (3168×4752)*, style* (2606×3176)*, SPST uses three scales; third row: content* (3024×4032)*, style* (3024×3787)*, SPST uses three scales; fifth row: content* (4480×5973)*, style* (6000×4747)*, SPST uses four scales. In comparison to our results, state of the art very fast methods produce images with many defects: halo effect, neural artifacts, blending, unfaithful color palette, ... This result in images that do not look like painting contrary to SPST outputs.*

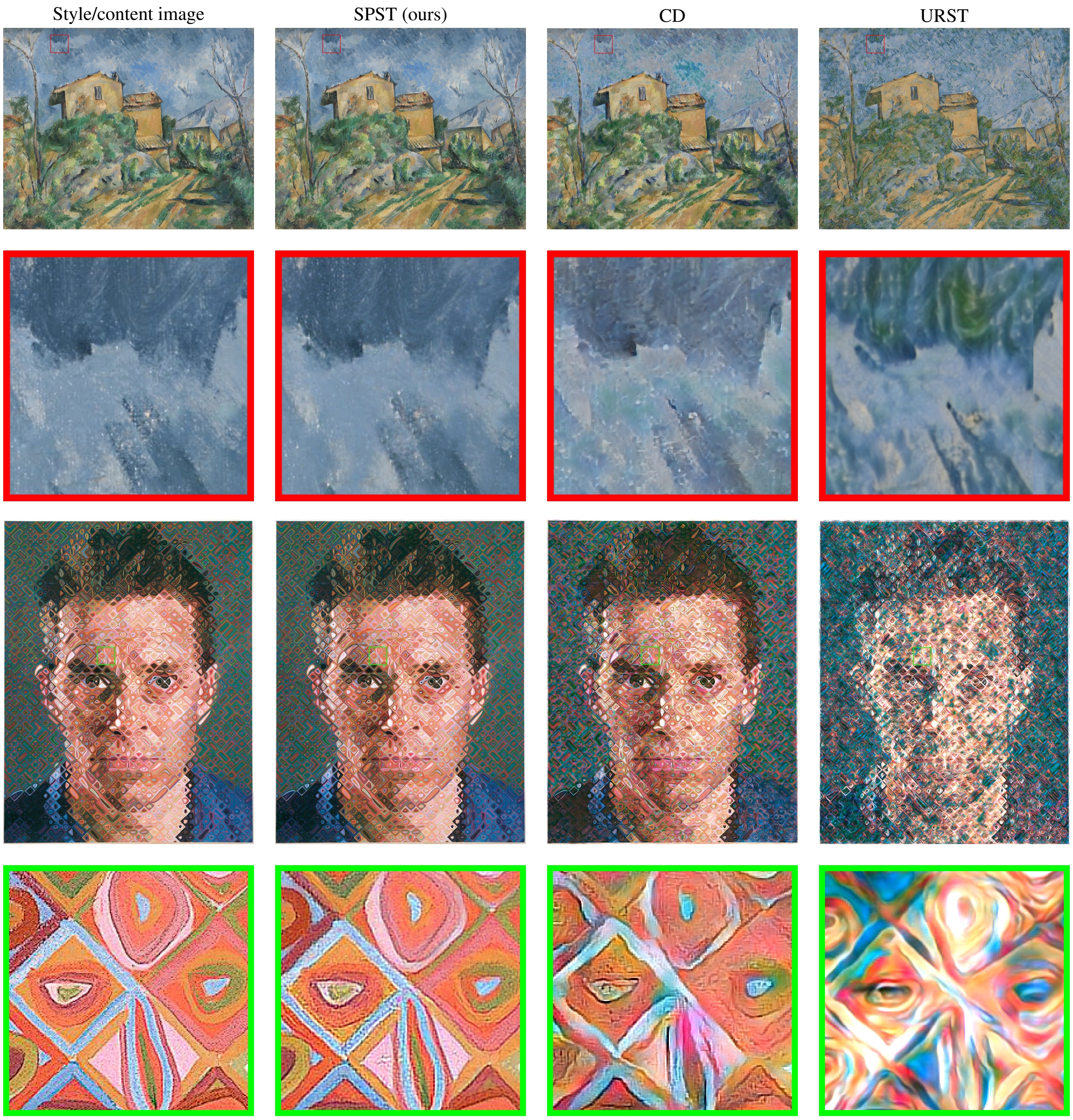

Figure 9: *Identity test: a style image is transferred to itself. We compare three style transfer strategies. From left to right: ground truth style, our result (SPST), CD [WLW*20], URST [CWX*22]. First row: The style image has resolution 3375×4201; Third row: The style image has resolution 3095×4000 (UHR images have been downscaled by ×4 factor to save memory). Second and fourth row: Close-up view with true resolution. Observe that our results are more faithful to the input painting and do not suffer from color blending.*

[JYF*20] JING, YONGCHENG, YANG, YEZHOU, FENG, ZUNLEI, et al. "Neural Style Transfer: A Review". *IEEE Transactions on Visualization and Computer Graphics* 26.11 (2020), 3365–3385. DOI: 10.1109/TVCG.2019.2921336 2.

[KLA*20] KARRAS, TERO, LAINE, SAMULI, AITTALA, MIIKA, et al. "Analyzing and improving the image quality of stylegan". *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, 8110–8119 4.

[LFY*17] LI, YIJUN, FANG, CHEN, YANG, JIMEI, et al. "Universal style transfer via feature transforms". *Advances in neural information processing systems* 30 (2017), 386–396 2–4, 8.

[LLC*21] LIN, CHIEH HUBERT, LEE, HSIN-YING, CHENG, YEN-CHI, et al. "InfinityGAN: Towards infinite-pixel image synthesis". *arXiv preprint arXiv:2104.03963* (2021) 4.

[LLKY19] LI, XUETING, LIU, SIFEI, KAUTZ, JAN, and YANG, MING-HSUAN. "Learning linear transformations for fast image and video style transfer". *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2019, 3809–3817 2, 4.

[LPSB17] LUAN, FUJUN, PARIS, SYLVAIN, SHECHTMAN, ELI, and BALA, KAVITA. "Deep Photo Style Transfer". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017 2.

[LW16] LI, CHUAN and WAND, MICHAEL. "Precomputed real-time texture synthesis with markovian generative adversarial networks". *European conference on computer vision*. Springer. 2016, 702–716 2.

[LWLH17] LI, YANGHAO, WANG, NAIYAN, LIU, JIAYING, and HOU, XIAODI. "Demystifying Neural Style Transfer". *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 2017, 2230–2236. DOI: 10.24963/ijcai.2017/310 4.

[LZW16] LU, YANG, ZHU, SONG-CHUN, and WU, YING NIAN. "Learning FRAME Models Using CNN Filters". *Thirtieth AAAI Conference on Artificial Intelligence*. 2016 2.

[Noc80] NOCEDAL, JORGE. "Updating Quasi-Newton Matrices with Limited Storage". *Mathematics of Computation* 35.151 (1980), 773–782. ISSN: 00255718, 10886842. DOI: 10.1090/S0025-5718-1980-0572855-7 4.

[PL19] PARK, DAE YOUNG and LEE, KWANG HEE. "Arbitrary style transfer with style-attentional networks". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, 5880–5888 3.

[RDB16] RUDER, MANUEL, DOSOVITSKIY, ALEXEY, and BROX, THOMAS. "Artistic style transfer for videos". *Pattern Recognition: 38th German Conference, GCPR 2016, Hannover, Germany, September 12-15, 2016, Proceedings 38*. Springer. 2016, 26–36 2.

[RWB17] RISSER, ERIC, WILMOT, PIERRE, and BARNES, CONNELLY. *Stable and Controllable Neural Texture Synthesis and Style Transfer Using Histogram Losses*. 2017. DOI: 10.48550/ARXIV.1701.08893 2, 4.

[SC17] SENDIK, OMRY and COHEN-OR, DANIEL. "Deep Correlations for Texture Synthesis". *ACM Trans. Graph.* 36.5 (July 2017). ISSN: 0730-0301. DOI: 10.1145/3015461 2, 4.

[SLJ*15] SZEGEDY, CHRISTIAN, LIU, WEI, JIA, YANGQING, et al. "Going Deeper With Convolutions". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015 4.

[SLSW18] SHENG, LU, LIN, ZIYI, SHAO, JING, and WANG, XIAOGANG. "Avatar-net: Multi-scale zero-shot style transfer by feature decoration". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, 8242–8250 3.

[Sne17] SNELGROVE, XAVIER. "High-resolution multi-scale neural texture synthesis". *SIGGRAPH Asia 2017 Technical Briefs*. 2017, 1–4 2.

[SZ15] SIMONYAN, KAREN and ZISSERMAN, ANDREW. "Very Deep Convolutional Networks for Large-Scale Image Recognition". *International Conference on Learning Representations*. 2015 2.

[TFF*20] TEXLER, ONDŘEJ, FUTSCHIK, DAVID, FIŠER, JAKUB, et al. "Arbitrary style transfer using neurally-guided patch-based synthesis". *Computers & Graphics* 87 (2020), 62–71. ISSN: 0097-8493. DOI: https://doi.org/10.1016/j.cag.2020.01.002 2, 4.

[ULVL16] ULYANOV, D., LEBEDEV, V., VEDALDI, A., and LEMPITSKY, V. "Texture Networks: Feed-forward Synthesis of Textures and Stylized Images". *ICML*. New York, NY, USA, 2016, 1349–1357. URL: http://dl.acm.org/citation.cfm?id=3045390.3045533 2, 3.

[UVL17] ULYANOV, DMITRY, VEDALDI, ANDREA, and LEMPITSKY, VICTOR. "Improved Texture Networks: Maximizing Quality and Diversity in Feed-Forward Stylization and Texture Synthesis". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017 3.

[VDKC20] VACHER, JONATHAN, DAVILA, AIDA, KOHN, ADAM, and COEN-CAGLI, RUBEN. "Texture Interpolation for Probing Visual Perception". *Advances in Neural Information Processing Systems*. Ed. by LAROCHELLE, H., RANZATO, M., HADSELL, R., et al. Vol. 33. Curran Associates, Inc., 2020, 22146–22157. URL: https://proceedings.neurips.cc/paper/2020/file/fba9d88164f3e2d9109ee770223212a0-Paper.pdf 2.

[WBSS04] WANG, ZHOU, BOVIK, A.C., SHEIKH, H.R., and SIMONCELLI, E.P. "Image quality assessment: from error visibility to structural similarity". *IEEE Transactions on Image Processing* 13.4 (2004), 600–612. DOI: 10.1109/TIP.2003.819861 10.

[WL00] WEI, L. Y. and LEVOY, M. "Fast texture synthesis using tree-structured vector quantization". *SIGGRAPH '00*. ACM Press/Addison-Wesley Publishing Co., 2000, 479–488. DOI: 10.1145/344779.345009 5.

[WLW*20] WANG, HUAN, LI, YIJUN, WANG, YUEHAI, et al. "Collaborative Distillation for Ultra-Resolution Universal Style Transfer". *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020 2, 4, 8, 10–13.

[WZZ*23] WANG, ZHIZHONG, ZHAO, LEI, ZUO, ZHIWEN, et al. "MicroAST: Towards Super-fast Ultra-Resolution Arbitrary Style Transfer". *Proceedings of the AAAI Conference on Artificial Intelligence* 37.3 (June 2023), 2742–2750. DOI: 10.1609/aaai.v37i3.25374 2, 4.

[ZIE*18] ZHANG, RICHARD, ISOLA, PHILLIP, EFROS, ALEXEI A., et al. "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018 10.

[ZKB*22] ZHANG, KAI, KOLKIN, NICK, BI, SAI, et al. "ARF: Artistic Radiance Fields". *ECCV 2022*. 2022 2.