Scaling Painting Style Transfer

Bruno Galerne, Lara Raad, José Lezama, Jean-Michel Morel

Scaling Painting Style Transfer

Bruno Galerne
Institut Denis Poisson
Université d'Orléans, Université de Tours, CNRS
Institut Universitaire de France (IUF)
bruno.galerne@univ-orleans.fr

Lara Raad
Laboratoire Informatique Gaspard Monge
Univ Gustave Eiffel, CNRS, LIGM
F-77454 Marne-la-Vallée, France
lara.raadcisa@esiee.fr

José Lezama*
Universidad de la República, Uruguay
Instituto de Ingenierá Eléctrica (IIE)
jlezama@fing.edu.uy

Jean-Michel Morel
Centre Borelli
ENS Paris-Saclay, University Paris-Saclay and CNRS
Gif-sur-Yvette, France
moreljeanmichel@gmail.com

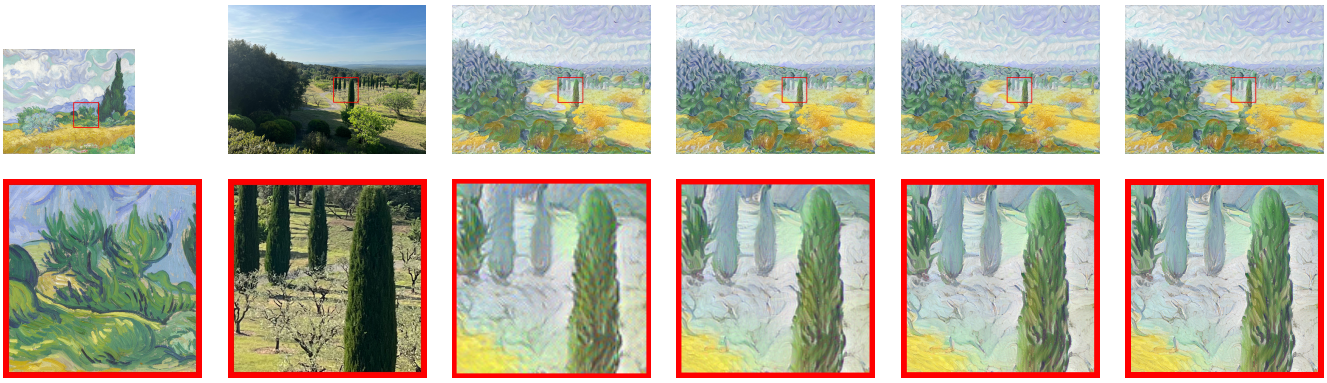

Figure 1. Ultra-high resolution multiscale style transfer. Top row from left to right: style (4226×5319), content (6048×8064), transfer at scale 1 (756×1008), 2 (1512×2016), 3 (3024×4032) and 4 (6048×8064). Bottom row: zoomed in detail for each image (from 128^2 to 1024^2). While the transfer is globally stable from one scale to the other, each upscaling allows to add fine pictural details which give an authentic painting aspect to the final output image. We recommend a screen examination of the images after ×8 zoom in.

Abstract

Neural style transfer is a deep learning technique that produces an unprecedentedly rich style transfer from a style image to a content image and is particularly impressive when it comes to transferring style from a painting to an image. It was originally achieved by solving an optimization problem to match the global style 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 neural style transfer and increase its resolution, but they all compromise the quality of the produced images. Indeed, transferring the style of a painting is a complex task involving features at different scales, from the colour 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 images, enabling multiscale style transfer 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 show that our method produces a style transfer of unmatched quality for such high resolution painting styles.

*José Lezama is now at Google Research. Contributed to this work while at Universidad de la República.

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. [2, 12]), it has seen a remarkable development since the seminal works of Gatys *et al*. [7, 8]. These works demonstrate that the Gram matrices of the activation functions of a pre-trained VGG19 network [26] faithfully encode the perceptual style and textures of an input image. Style transfer is performed by optimizing a functional aiming at a compromise between fidelity to VGG19 features of the content image while reproducing the Gram matrix statistics of the style image. Other global statistics have been proven effective for style transfer and texture synthesis [6, 10, 11, 19, 20, 23, 24, 31] 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) [9, 10, 27]. The two major drawbacks of such optimization-based style transfer are the computation time and the limited resolution of images because of large GPU memory requirement.

Regarding computation time, several methods have been proposed to generate new stylized images by training feed-forward networks [14, 16, 29] or by training VGG encoder-decoder networks [3, 5, 13, 17, 18]. These models tend to provide images with relatively low style transfer loss and can therefore be considered as approximate solutions to [8]. Despite remarkable progress regarding computation time, 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 was recently tackled [1, 4, 32]. Nevertheless, although generating ultra-high resolution (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) [4]. In other methods, the multiscale nature of the networks is not fully exploited [32].

As illustrated in Figure 1, our high-resolution multiscale method manages to transfer the different levels of detail contained in the style image from the colour 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 illustrate that the results of competitive methods suffer from brushstroke styles that do not match those of the UHR style image, and that very fine textures are not well transferred 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.

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 this algorithm allows a multi-resolution UHR transfer for images up to 8196^2 px in size.
- We experimentally show that the visual quality of this UHR style transfer is richer and more faithful than recent fast but approximate solutions.

This work provides a new reference method for high-quality style transfer with unequaled multi-resolution depth. It might serve as a reference to evaluate fast but approximate models.

2. Related work

Style transfer by optimization. As recalled in the introduction, the seminal work of Gatys *et al*. formulated style transfer as an optimization minimizing the distances between Gram matrices of VGG features. Other global statistics have been proven effective for style transfer and texture synthesis such as deep correlations [10, 24], Bures metric [31], spatial mean of features [6, 19], feature histograms [23], or even the full feature distributions [11]. Specific cost function corrections have also been proposed for photorealistic style transfer [20]. 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 [9, 10, 27].

Style transfer by training feed-forward networks. Ulyanov *et al*. [29, 30] and Johnson *et al*. [14] showed that one could train a feed-forward network to approximately solve style transfer. Although these models produce a very fast style transfer, they require learning a new model for each type of style.

Universal style transfer (UST). Style limitation has been addressed by training a VGG autoencoder that attempts to reverse VGG feature computations after normalizing them at the autoencoder bottleneck. Chen *et al*. [3] introduce the encoder-decoder framework with a style swap layer replacing content features with the closest style features on overlapping patches. Huang *et al*. [13] propose to use an Adaptive Instance Normalization (AdaIN) that adjusts the mean and variance of the content image features to match those of

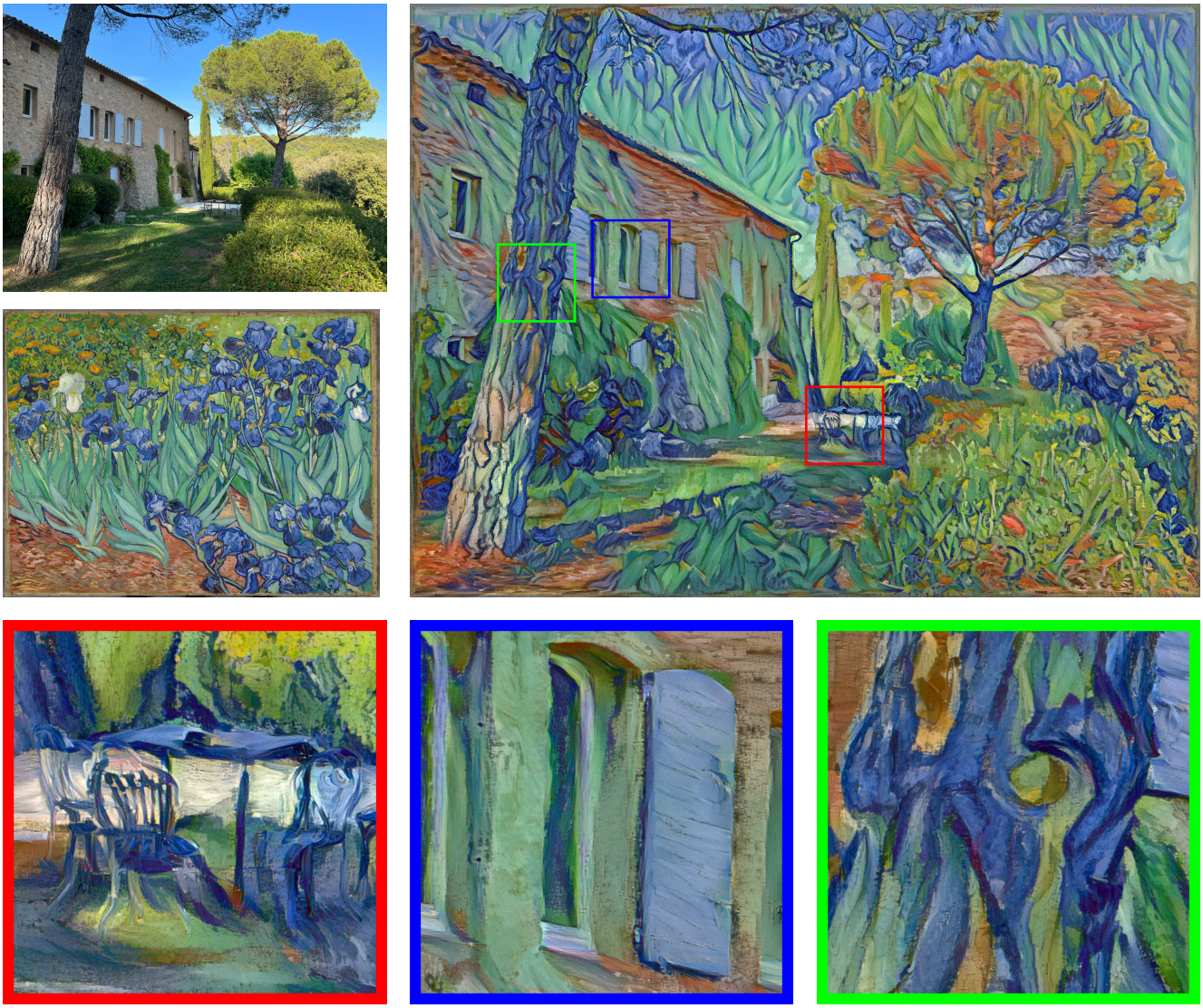

Figure 2. UHR style transfer. Top row, content image (top-left, 6048×8064), style image (bottom left, 6048×7914), result (right, 6048×8064) (the three UHR images are downscaled ×4 for visualization). Bottom row: three zoomed in details of the result image (800^2, true resolution). Observe how very fine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

the style image. Li *et al*. [18] 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*. [25] use one encoder-decoder block combining the transformations of [18] and [3]. Park *et al*. [22] introduce an attention-based transformation module to integrate the local style patterns according to the spatial distribution of the content image. Li *et al*. [17] train a symmetric encoder-decoder image reconstruction module and a transformation learning module. Chiu *et al*. [5] extend [18] by embedding a new transformation that iteratively updates features in the cascade of four autoencoder modules. Despite the numerous improvements of fast UST strategies, let us remark that: (a) they rely on matching VGG statistics as introduced by Gatys *et al*. [8] (b) they are limited in resolution due to GPU memory required for the large sized models.

UST for high-resolution images. Some methods attempt to reduce the size of the network in order to perform high resolution style transfer. An *et al*. [1] propose ArtNet which is a channel-wise pruned version of GoogLeNet [28]. Wang *et al*. [32] 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 re-

ducing the number of convolutional filters involved in [18] and [13]. Chen *et al*. [4] recently 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.

3. Global optimization for neural style transfer

Single scale style transfer. Let us recall the algorithm of Gatys *et al*. [8]. 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

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

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

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

with

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

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

$$\begin{aligned} G^L(w) &= \frac{1}{n^L_\mathrm{p}} V^L(w)^\top V^L(w) \\ &= \frac{1}{n^L_\mathrm{p}} \sum_{k=0}^{n^L_\mathrm{p}} V^L(w)_k (V^L(w)_k)^\top \in \mathbb{R}^{n^L_\mathrm{c} \times n^L_\mathrm{c}}, \end{aligned} \quad (4)$$

where $V^L(w)_k \in \mathbb{R}^{n^L_\mathrm{c}}$ is the column vector corresponding to the k-th line of $V^L(w)$. $E^L_\mathrm{style}(x;v)$ is a fourth-degree polynomial and non convex with respect to (wrt) the VGG features $V^L(x)$. Gatys *et al*. [7] propose to use the L-BFGS algorithm [21] 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. The history size is typically 100 but will be decreased to 10 for HR images (for all scales except the first one) to limit memory requirement.

Gram loss correction. It is known that optimizing for the Gram matrix alone may introduce some loss of contrast artefacts since Gram matrices encompass information regarding both the mean values and correlation of features [11, 23, 24]. Instead of considering the full histogram of the features [11, 23], we found that correcting for the mean and standard deviation (std) of each feature gives visually satisfying results. Given some (reshaped) features $V \in \mathbb{R}^{n_\mathrm{p} \times n_\mathrm{c}}$, define $\mathrm{mean}(V)$ and $\mathrm{std}(V) \in \mathbb{R}^{n_\mathrm{c}}$ by

$$\mathrm{mean}(V)_j = \frac{1}{n_\mathrm{p}} \sum_{k=1}^{n_\mathrm{p}} V_{k,j} \quad (5)$$

and

$$\mathrm{std}(V)_j = \Big(\frac{1}{n_\mathrm{p}} \sum_{k=1}^{n_\mathrm{p}} (V_{k,j} - (\mathrm{mean}(V))_j)^2 \Big)^{\frac{1}{2}}, \quad (6)$$

$j \in \{1, \dots, n_\mathrm{c}\}$. In the whole paper, we replace the Gram loss $w_L \left\| G^L(u) - G^L(v) \right\|^2_\mathrm{F}$ of Eq. (3) by the following augmented style loss

$$\begin{aligned} \tilde{E}^L_\mathrm{style}(x;v) =& w_L \left\| G^L(u) - G^L(v) \right\|^2_\mathrm{F} \\ &+ w'_L \| \mathrm{mean}(V^L(x)) - \mathrm{mean}(V^L(v)) \|^2 \\ &+ w''_L \| \mathrm{std}(V^L(x)) - \mathrm{std}(V^L(v)) \|^2 \end{aligned} \quad (7)$$

for a better reproduction of the feature distribution. The values of all the weights λ_c, w_L, w'_L, w''_L, $L \in \mathcal{L}_\mathrm{s}$, have been fixed for all images. Note that limiting our style loss $\tilde{E}^L_\mathrm{style}(x;v)$ to second-order statistics is capital for a straightforward implementation of our localized algorithm described in Section 4.

Multiscale style transfer. Since the style transfer solely relies on VGG19, the transfer is spatially limited by the receptive field of the network [9]. For images having a side larger than 500 px, visually richer results are obtained by adopting a multiscale approach [9] corresponding to the standard coarse-to-fine approach for texture synthesis [34]. For the sake of simplicity, suppose that the content image u and the style image v have the same resolution (otherwise one can downscale the resolution of v to match the one of u as a preprocessing [8]). When using $n_\mathrm{scales} > 1$, both u and v are first downscaled by a factor $2^{n_\mathrm{scales}-1}$ to obtain the low-resolution couple $(u^\downarrow, v^\downarrow)$ and style transfer is first applied at this coarse resolution starting with $x_0 = u^\downarrow$. Then, for each subsequent scale $s = 2$ to n_scales, the result image $x^\downarrow$ is upscaled by a factor 2 to define the initialization image x_0, and style transfer is applied with the content and style image downscaled by a factor $2^{n_\mathrm{scales}-s}$. At the last scale the output image has the same resolution as the HR content image. Thanks to this coarse-to-fine approach, the style is

transferred in a coarse-to-fine way. This is especially important when using an HR digital photograph of a painting for the style: Ideally, the first scale encompasses color and large strokes while subsequent scales refine the stroke details up to the painting texture, bristle brushes and canvas texture.

Unfortunately, applying this multiscale 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 neural features and style transfer loss gradient

Our main contribution is to emulate the computation of $\nabla_x E_{\text{transfer}}(x;(u,v))$ even for images larger than 4000^2 px for which evaluation and automatic differentiation of the loss is not feasible due to large memory requirement.

First suppose one wants to compute the feature maps $V^L(x)$, $L \in \mathcal{L}_{\text{s}} \cup \{L_{\text{c}}\}$, 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 $E_{\text{transfer}}(x;(u,v))$ (1). Note that it is not possible to automatically differentiate this loss, because the computation graph linking back to x is lost.

However, a close inspection of the different style losses wrt the neural features shows that they all have the same form: For each style layer $L \in \mathcal{L}_{\text{s}}$, the gradient of the layer style loss $\tilde{E}^L_{\text{style}}(x;v)$ wrt the layer feature $V^L(x)_k \in \mathbb{R}^{n^L_{\text{c}}}$ 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 $V^L(v)$. This fact is summarized in the formulas of Table 1. 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.

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

Input: Current image x, content image layer $V^{L_{\text{c}}}(u)$, and list of feature statistics of v $\{(G^L(v), \text{mean}(V^L(v)), \text{std}(V^L(v))),\ L \in \mathcal{L}_{\text{s}}\}$ (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 $\text{VGG}(b)$ without computation graph

For each style layer $L \in \mathcal{L}_{\text{s}}$: Extract the features of the block by properly removing the margin and add their contribution to $G^L(x)$ and $\text{mean}(V^L(x))$.

end for

For each style layer $L \in \mathcal{L}_{\text{s}}$: compute $\text{std}(V^L(x))$ as a function of $G^L(x)$ and $\text{mean}(V^L(x))$.

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 $\text{VGG}(b)$ with computation graph

For each style layer $L \in \mathcal{L}_{\text{s}}$: 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 $V^{L_{\text{c}}}(b)$ 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

Note that the memory requirement for Algorithm 1 does not depend on the image size. Indeed, by spatially splitting all the computations involving VGG19 features, dealing with larger images only requires more computation time

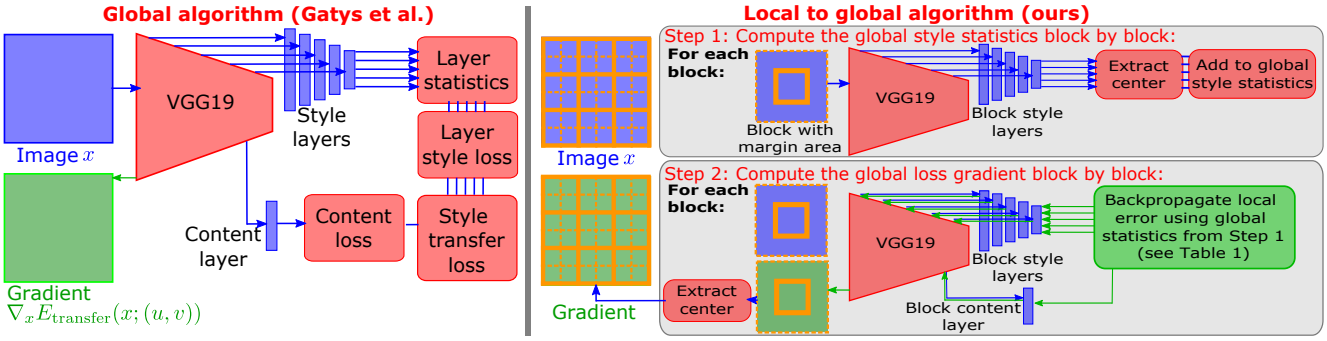

Figure 3. Algorithm overview: Our localized algorithm (right part) allows to compute the global style transfer loss and its gradient wrt x for images that are too large for the original algorithm of Gatys *et al.* [8] (left part).

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: $G = \frac{1}{n_{\text{p}}} VV^T$	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 (having size $n_{\text{p}} \times n_{\text{c}}$).

(since there are more blocks). However, the L-BFGS optimization will require more memory since it requires to store a gradients history, each gradient having the size of x. Using a single 40 GB GPU, our algorithm allows for style transfer for images of size up to 8192^2 px.

5. Experiments

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 [8] (except for our slightly modified style loss) and suffers from poor image resolution and grid artefacts. Note how, 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.

We compare our method with two fast alternatives for UHR style transfer, namely collaborative distillation (CD) [32] and URST [4] (based on [18]) using their official implementations[1]. 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

[1] [32]: https://github.com/MingSun-Tse/Collaborative-Distillation; [4]: https://git.io/URST

Method	PSNR ↑	SSIM ↑	LPIPS ↓	Gram ↓
SPST (ours)	**24.6**	**0.457**	**0.352**	**1.90e5**
CD	21.8	0.417	0.501	4.23e7
URST	19.0	0.415	0.545	6.53e7

Table 2. Quantitative evaluation of *identity test* for UHR style transfer. The PSNR, SSIM [33], LPIPS [35] and the Gram (*style distance*) metrics are shown for our results (SPST), CD [32] and URST [4]. All metrics are averages using 79 HR paintings images used as both content and style. Best results shown in bold.

and fine details cannot be transferred. 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 4 that our method is the only one capable of conveying the aspect of the painting strokes to the content image. CD suffers from halo and high-frequency artefacts, while URST presents visible patch boundaries and a detail frequency mismatch due to improper scaling. Observe also 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.

Identity test for style transfer quality assessment. Style transfer is an ill-posed problem by nature. We introduce

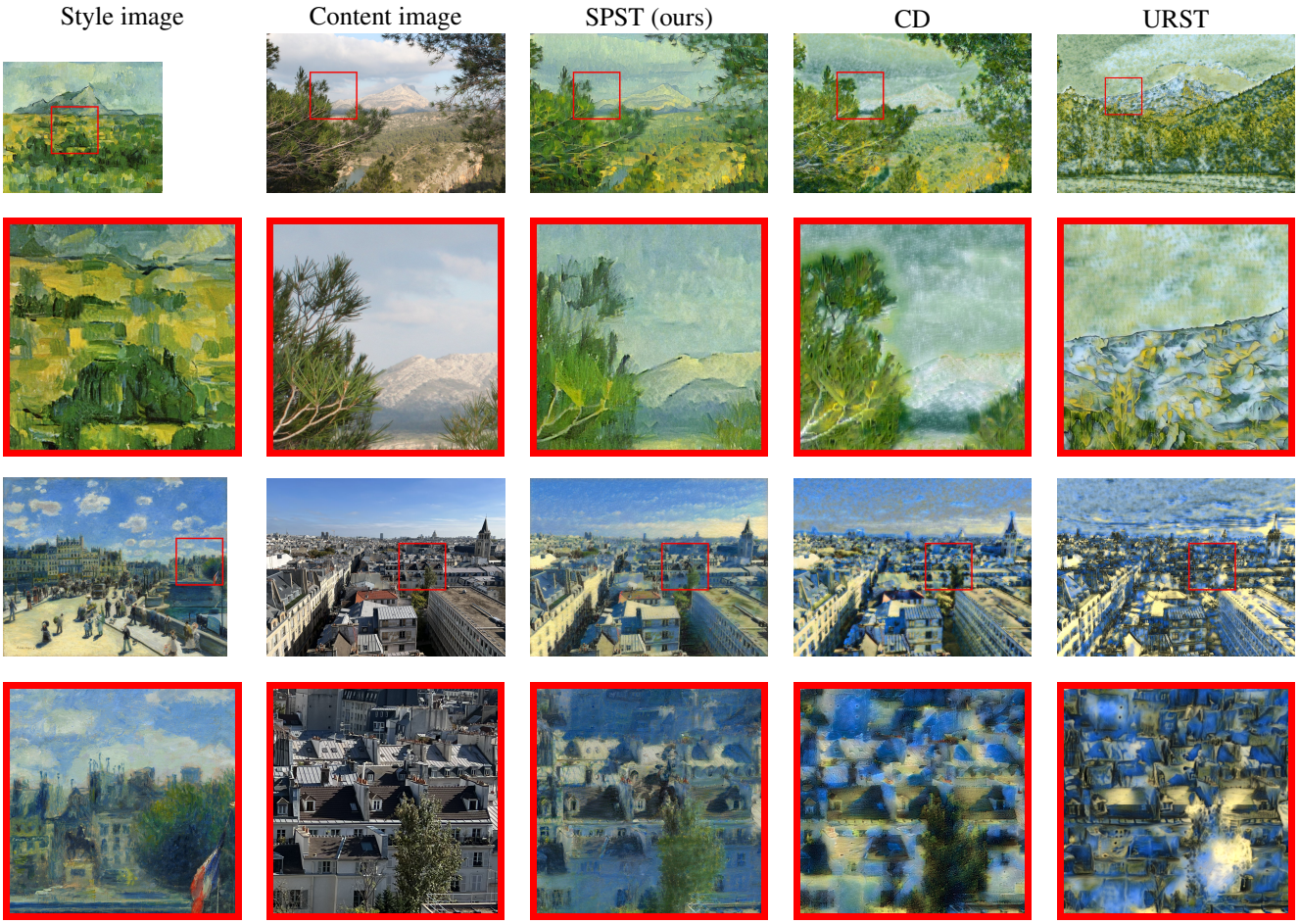

Figure 4. Comparison of UHR style transfers. For each example, top row, left to right: style, content, our result (SPST), CD [32], URST [4]. Bottom row: zoom in of the corresponding top row. First row: content (3168×4752), style (2606×3176). Third row: content (3024×4032), style (3024×3787). We used three scales for both of our results. Observe the loss of details and the unrealistic looks of the outputs produced by both fast methods.

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 5. We observe that our multiscale 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 [32] suffer from color deviation and frequency artefacts while the results of [4] apply a style transfer that is too homogeneous and present color and scale issues as already discussed. Some previous work introduce a *style distance* [32] 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 calculate PSNR, SSIM [33] and LPIPS [35] metrics on a set of 79 paint styles (see supplementary material) to quantitatively evaluate our results, in addition to the "Gram" metric, that is, the style loss of Equation (8) using the original Gram loss of Equation (3), computed on UHR results using our localized approach. The average scores reported in Table 2 confirm the good qualitative behaviour discussed earlier: Our method is by far the best for all the scores.

Computation time. Our UHR style transfer algorithm takes several minutes, from 16 minutes for the third row result in Figure 4 to 74 minutes for the result in Figure 2 using an A100 GPU. In comparison, fast UST methods only take a few seconds.

6. Discussion

Our work presented an extension of the Gatys *et al*. style transfer algorithm to UHR images. Regarding visual quality, our algorithm clearly outperforms competing UHR

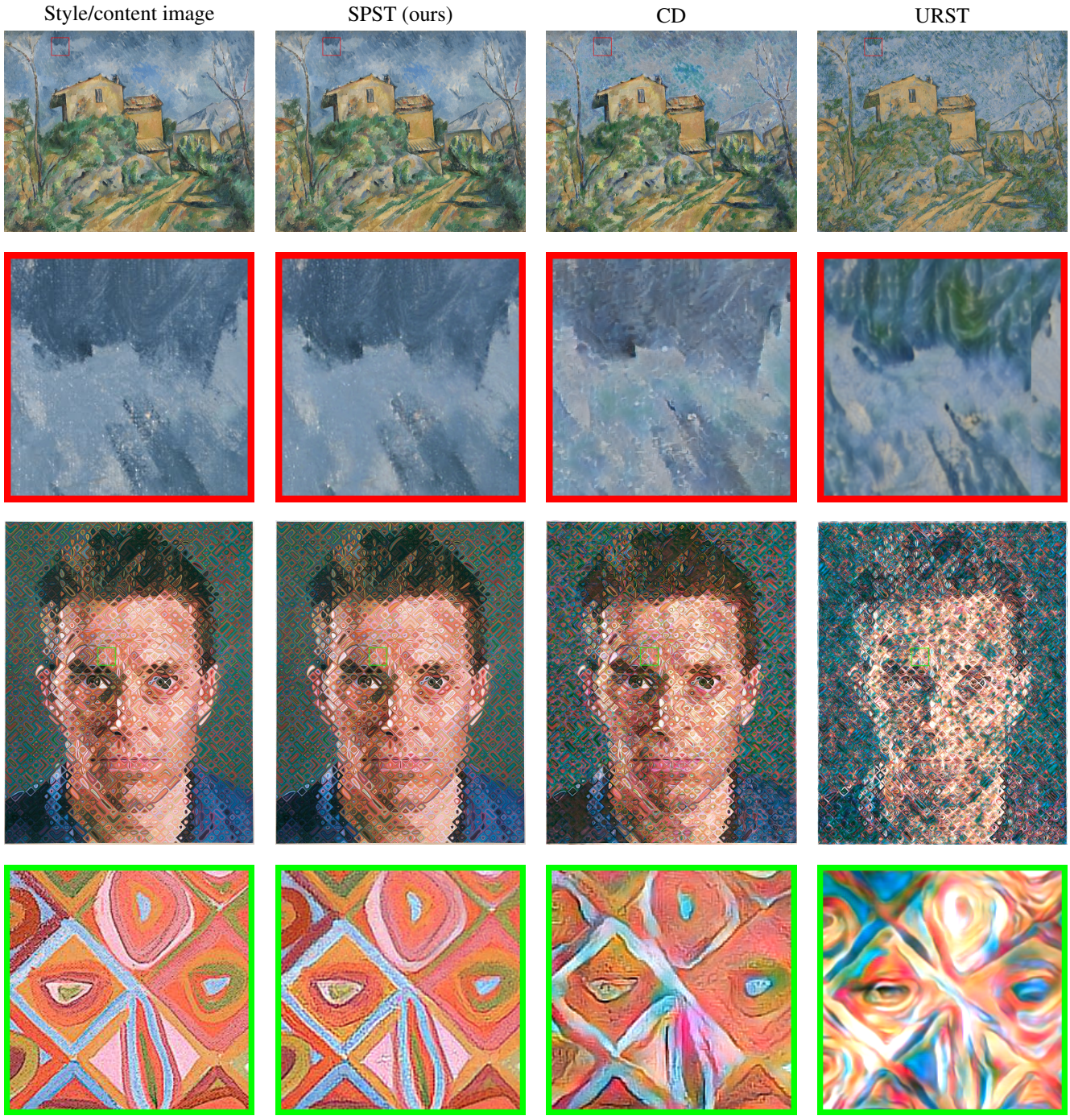

Figure 5. Identity test: a style image is transferred to itself. We compare three style transfer strategies. From left to right: ground truth style, our result (SPST), CD [32], URST [4]. First row: The style image has resolution 3375×4201; Third row: The style image has resolution 3095×4000 (UHR images have been downscaled by ×4 factor to save memory). Second and fourth row: Close-up view with true resolution. Observe that our results are the more faithful to the input painting and do not suffer from color blending

methods by conveying a true painting feel thanks to faithful HR details such as strokes, paint cracks, and canvas texture.

It is here that we may confess: Our iterative method is obviously slow, even though its complexity scales linearly with image size. Yet, as we have demonstrated, fast methods do not reach a satisfying quality, and fast high-quality style transfer remains an open problem to this date.

Several extensions and applications of our work can be

considered. For instance, we can perform HR texture synthesis by removing the content term [7] (see supplementary material). Our two-pass procedure can be extended to any function of the Gram matrix and feature spatial means, such as the Bures metric used in [31] for texture mixing. One could also consider extending the method to the slice Wasserstein style loss [11] using the Run-Sort-ReRun strategy [15]. However, the memory requirements to store five VGG feature maps (or their projections) increase linearly with the size of the input image, in contrast to the size-agnostic global statistics used in this paper.

This work opens the way for several future research directions, from allowing local control for UHR style transfer [9] to training fast CNN-based models to reproduce our results.

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

References

[1] Jie An, Tao Li, Haozhi Huang, Li Shen, Xuan Wang, Yongyi Tang, Jinwen Ma, Wei Liu, and Jiebo Luo. Real-time universal style transfer on high-resolution images via zero-channel pruning. *arXiv preprint arXiv:2006.09029*, 2020. 2, 3

[2] Mathieu Aubry, Sylvain Paris, Samuel W. Hasinoff, Jan Kautz, and Frédo Durand. Fast local laplacian filters: Theory and applications. *ACM Trans. Graph.*, 33(5), sep 2014. 2

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

[4] Zhe Chen, Wenhai Wang, Enze Xie, Tong Lu, and Ping Luo. Towards ultra-resolution neural style transfer via thumbnail instance normalization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 393–400, Jun. 2022. 2, 4, 6, 7, 8, 11, 13, 14

[5] Tai-Yin Chiu and Danna Gurari. Iterative feature transformation for fast and versatile universal style transfer. In *European Conference on Computer Vision*, pages 169–184. Springer, 2020. 2, 3

[6] Valentin De Bortoli, Agnès Desolneux, Alain Durmus, Bruno Galerne, and Arthur Leclaire. Maximum entropy methods for texture synthesis: Theory and practice. *SIAM Journal on Mathematics of Data Science*, 3(1):52–82, 2021. 2

[7] L. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems 28*, pages 262 – 270. 2015. 2, 4, 9, 11

[8] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, June 2016. 2, 3, 4, 6

[9] Leon A. Gatys, Alexander S. Ecker, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Controlling perceptual factors in neural style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2, 4, 9, 11

[10] Nicolas Gonthier, Yann Gousseau, and Saïd Ladjal. High-resolution neural texture synthesis with long-range constraints. *Journal of Mathematical Imaging and Vision*, 64(5):478–492, 2022. 2, 11

[11] Eric Heitz, Kenneth Vanhoey, Thomas Chambon, and Laurent Belcour. A sliced wasserstein loss for neural texture synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9412–9420, June 2021. 2, 4, 9

[12] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, page 327–340, New York, NY, USA, 2001. Association for Computing Machinery. 2

[13] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2, 4

[14] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711, 2016. 2

[15] José Lezama, Wei Chen, and Qiang Qiu. Run-sort-rerun: Escaping batch size limitations in sliced wasserstein generative models. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 6275–6285. PMLR, 18–24 Jul 2021. 9

[16] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European conference on computer vision*, pages 702–716. Springer, 2016. 2

[17] Xueting Li, Sifei Liu, Jan Kautz, and Ming-Hsuan Yang. Learning linear transformations for fast image and video style transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3809–3817, 2019. 2, 3

[18] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. *Advances in neural information processing systems*, 30:386–396, 2017. 2, 3, 4, 6

[19] Yang Lu, Song-chun Zhu, and Ying Nian Wu. Learning frame models using cnn filters. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016. 2

[20] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[21] Jorge Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980. 4

[22] Dae Young Park and Kwang Hee Lee. Arbitrary style transfer with style-attentional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5880–5888, 2019. 3

[23] Eric Risser, Pierre Wilmot, and Connelly Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses, 2017. 2, 4

[24] Omry Sendik and Daniel Cohen-Or. Deep correlations for texture synthesis. *ACM Trans. Graph.*, 36(5), jul 2017. 2, 4

[25] Lu Sheng, Ziyi Lin, Jing Shao, and Xiaogang Wang. Avatar-net: Multi-scale zero-shot style transfer by feature decoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8242–8250, 2018. 3

[26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 2

[27] Xavier Snelgrove. High-resolution multi-scale neural texture synthesis. In *SIGGRAPH Asia 2017 Technical Briefs*, pages 1–4. 2017. 2

[28] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 3

[29] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, pages 1349–1357, 2016. 2

[30] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[31] Jonathan Vacher, Aida Davila, Adam Kohn, and Ruben Coen-Cagli. Texture interpolation for probing visual perception. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22146–22157. Curran Associates, Inc., 2020. 2, 9

[32] Huan Wang, Yijun Li, Yuehai Wang, Haoji Hu, and Ming-Hsuan Yang. Collaborative distillation for ultra-resolution universal style transfer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2, 3, 6, 7, 8, 11, 13, 14

[33] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 6, 7

[34] L. Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH '00*, pages 479–488. ACM Press/Addison-Wesley Publishing Co., 2000. 4

[35] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 6, 7

Supplementary material

This supplementary material presents additional style transfer results, gives an overview of the dataset of pictures used for the *identity test* comparison, and discuss the adaptation of the algorithm for UHR texture synthesis.

A. Additional style transfer results

A.1. Multiscale examples

Figure 6 presents a second example of ultra-high resolution (UHR) style tranfer with each intermediate results of the multiscale algorithm. One can see how the texture of the painting and the stroke details are gradually refined.

A.2. Additional comparison experiments

Figure 7 presents some additional comparison examples. These results show that the competing methods suffer from color imbalance and do not match the fine texture and strokes of the style painting. This is due to the fact that neither URST [4] nor CD [32] take into account details at different scales. On the other hand, we are able to convey the appearance of fine details and painting strokes to the content image.

Figure 8 shows some examples of style transfer on portraits. Once again, we can observe that our method is the only one able to transfer the texture and strokes of the painting to the portrait content.

Figure 9 shows some examples of failure cases that underline the importance of correctly choosing the couple content/style for a correct style transfer. The first row example uses the same style as in Figure 8 but with a different content image where the proportion of the face space is much larger than in the style image. Note also that the lack of a beard in the content produces an undesirable effect on the face. The second row example lack of details and the style transfer creates undesirable noisy patterns in the sky and grass. As said in the conclusion of the main paper, allowing local control [9] for UHR style transfer is an interesting direction for future developments.

A.3. Full resolution images

In the main paper all UHR images have been downscaled by a factor $\times 4$, with highlighted details included with the true resolution. In the following figures, we include the style and content images with a better resolution (only downgraded by a factor $\times 2$) and our style transfer result in full resolution. Note that images have been compressed using jpeg quality 85 to limit the .pdf file size. See Figure 10, 11, 12, and 13.

B. Painting images dataset for the *identity test* experiment

Figure 14 shows the 79 UHR painting images used for the *identity test* where we evaluate whether a method was able to reproduce a painting when the content and style image were identical.

C. Texture synthesis

As said in the Discussion section of the paper, our approach also allows for UHR texture synthesis. Following the original paper on texture synthesis [7], given a texture exemplar v, texture synthesis is performed by minimizing

$$E_{\text{style}}(x;v) = \sum_{L\in\mathcal{L}_\text{s}} E^L_{\text{style}}(x;v) \tag{8}$$

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 following differences:

- The style image is replaced by the texture image.
- There is no content image and no content loss (set $\lambda_\text{c} = 0$).
- The image x is initialized by a random noise x_0.

We perform texture synthesis following the same multiscale approach and using the using our augmented style loss

$$\begin{aligned}\tilde{E}^L_{\text{style}}(x;v) =& w_L \left\|G^L(u) - G^L(v)\right\|^2_\text{F} \\ &+ w'_L \|\operatorname{mean}(V^L(x)) - \operatorname{mean}(V^L(v))\|^2 \\ &+ w''_L \|\operatorname{std}(V^L(x)) - \operatorname{std}(V^L(v))\|^2.\end{aligned} \tag{9}$$

However, for texture synthesis starting with a random image, the number of scales should be as high as possible, that is the multiscale process starts with images of moderate size (about 200 pixels). To illustrate this point we show eight different texture synthesis in Figures 15, 16, 17 and 18. For each example we show the synthesis using 3 scales (same setting as for style transfer) and 5 scales. We remark that it is quite important to start with a first scale with small size for a satisfying synthesis quality that reproduces the texture statistics. Using only 3 scales yields textures that are spatially homogeneous due to the white noise initialization. Let us remark that to the best of our knowledge the highest resolution produced in the literature was limited to 2048^2 pixels [10] for the multiscale version of Gatys *et al*. algorithm [7].

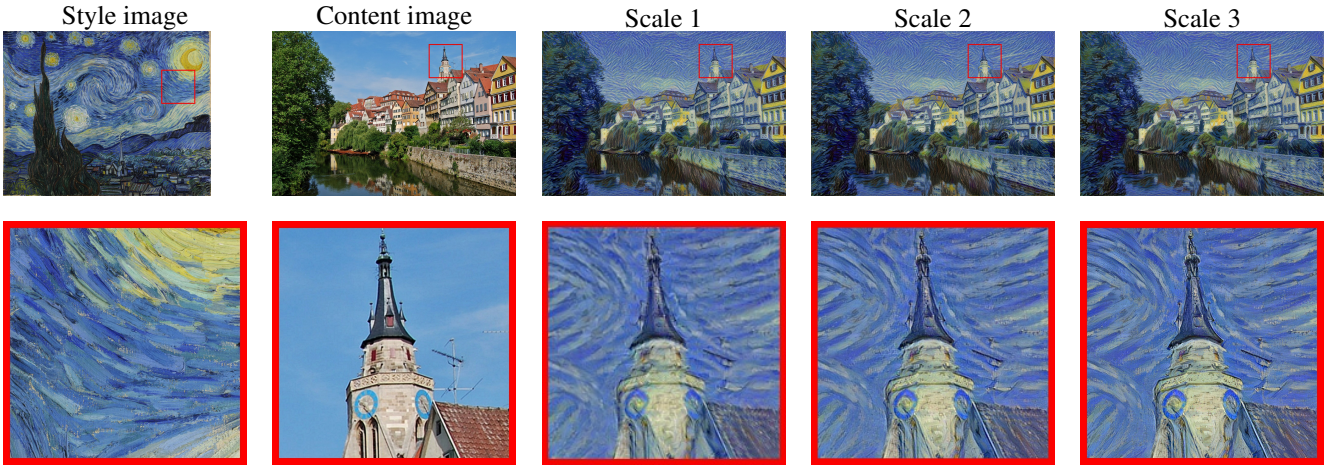

Figure 6. UHR multiscale style transfer. Top row from left to right: style (3906×4933), content (4933×5800), transfer at scale 1 (976×1450), 2 (1953×2900), 3 (4933×5800). Bottom row: zoomed in detail for each image (from 200^2 to 800^2). While the transfer is globally stable from one scale to the other, each upscaling allows to add fine pictural details which give an authentic painting aspect to the final output image. We recommend a screen examination of the images after ×8 zoom in.

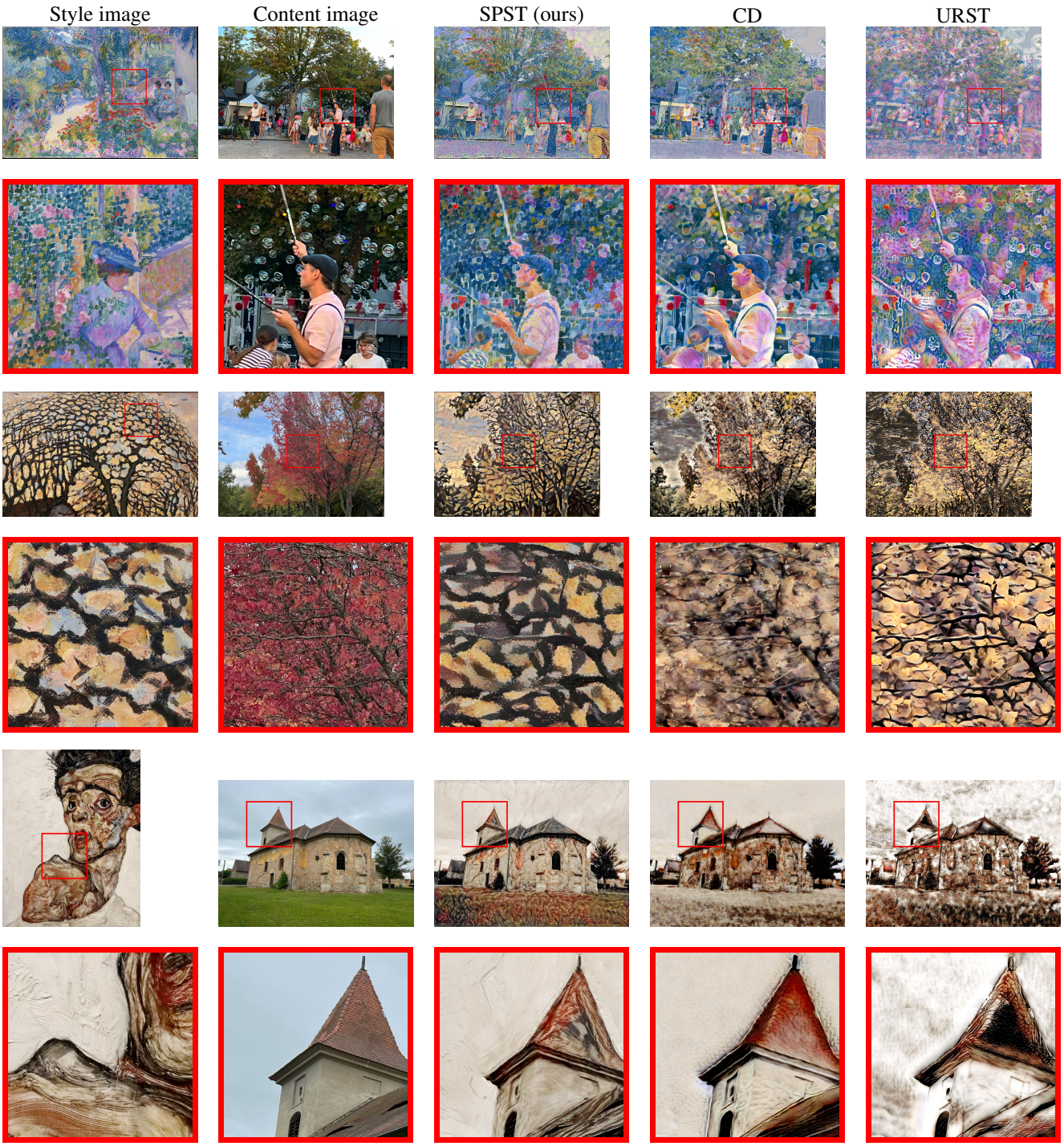

Figure 7. Comparison of UHR style transfers. For each example, top row, left to right: style, content, our result (SPST), CD [32], URST [4]. Bottom row: zoom in of the corresponding top row. First row: content (3024×4032), style (3024×4477). Third row: content (3024×4032), style (3024×4738). Fifth row: content (3024×4032), style (3655×2836). We used three scales for all our results. Observe the loss of details and the unrealistic looks of the outputs produced by both fast methods.

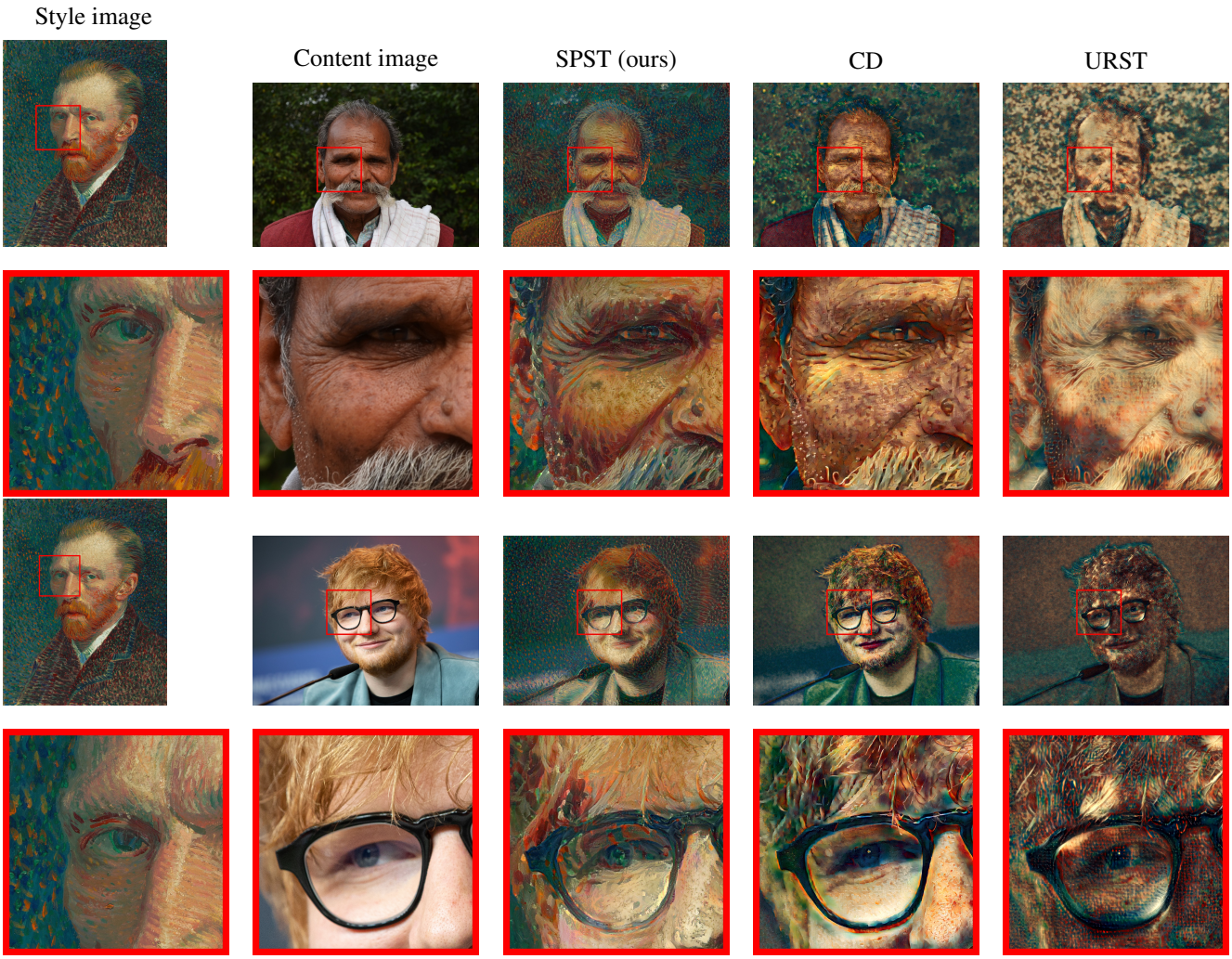

Figure 8. Comparison of UHR style transfer on portraits. For each example, top row, left to right: style, content, our result (SPST), CD [32], URST [4]. Bottom row: zoom in of the corresponding top row. First row: content (3264×4512), style (4126×3264). Third row: content (4480×5973), style (6000×4747). We used three scales for our result in the first example and four scales for our result in the second example. Observe the loss of details and the unrealistic looks of the outputs produced by both fast methods.

Figure 9. Examples of UHR style transfer failure cases. For each example, from left to right: style, content and our result. First row: content (5184×3456), style (4368×3456). Second row: content (3024×4032), style (3024×3787). We used three scales for both results.

Figure 10. UHR style transfer in full resolution (from Figure 1 of the main paper). Top row: style image (4226×5319), content image (6048×8064). Bottom: result (6048×8064). Observe how very fine details such as the chairs look as if painted.

Figure 11. UHR style transfer in full resolution (from Figure 2 of the main paper). Top row: style image (6048×7914), content image (6048×8064). Bottom: result (6048×8064). Observe how very fine details such as the chairs look as if painted.

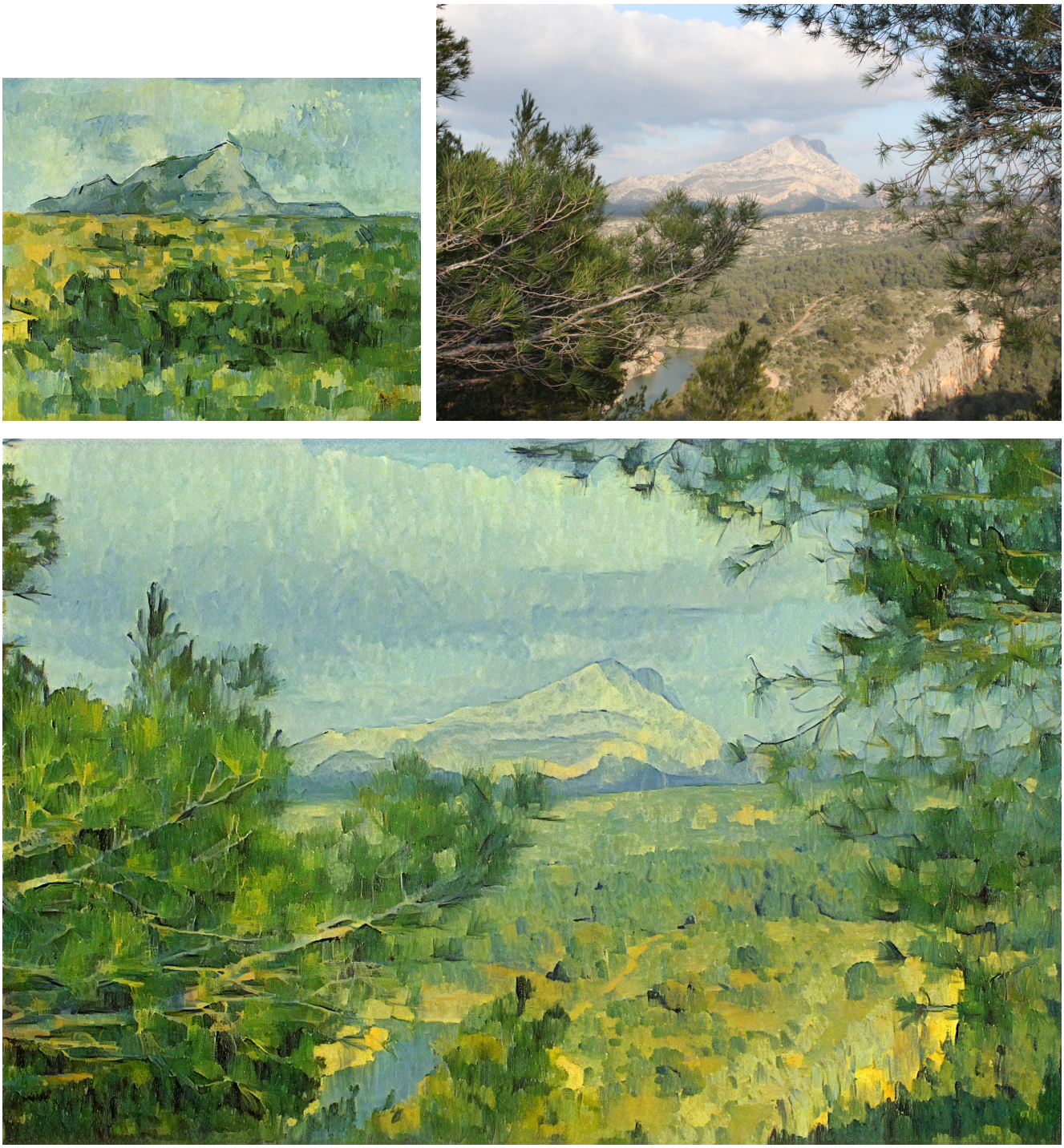

Figure 12. UHR style transfer in full resolution (from Figure 4 of the main paper). Top row: style image (2606×3176), content image (3168×4752). Bottom: result (3168×4752).

Figure 13. UHR style transfer in full resolution (from Figure 4 of the main paper). Top row: style image (3024×3787), content image (3024×4032). Bottom: result (3024×4032).

Figure 14. Overview of the 79 UHR painting images used for the *identity test*.

Figure 15. UHR texture synthesis: 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)

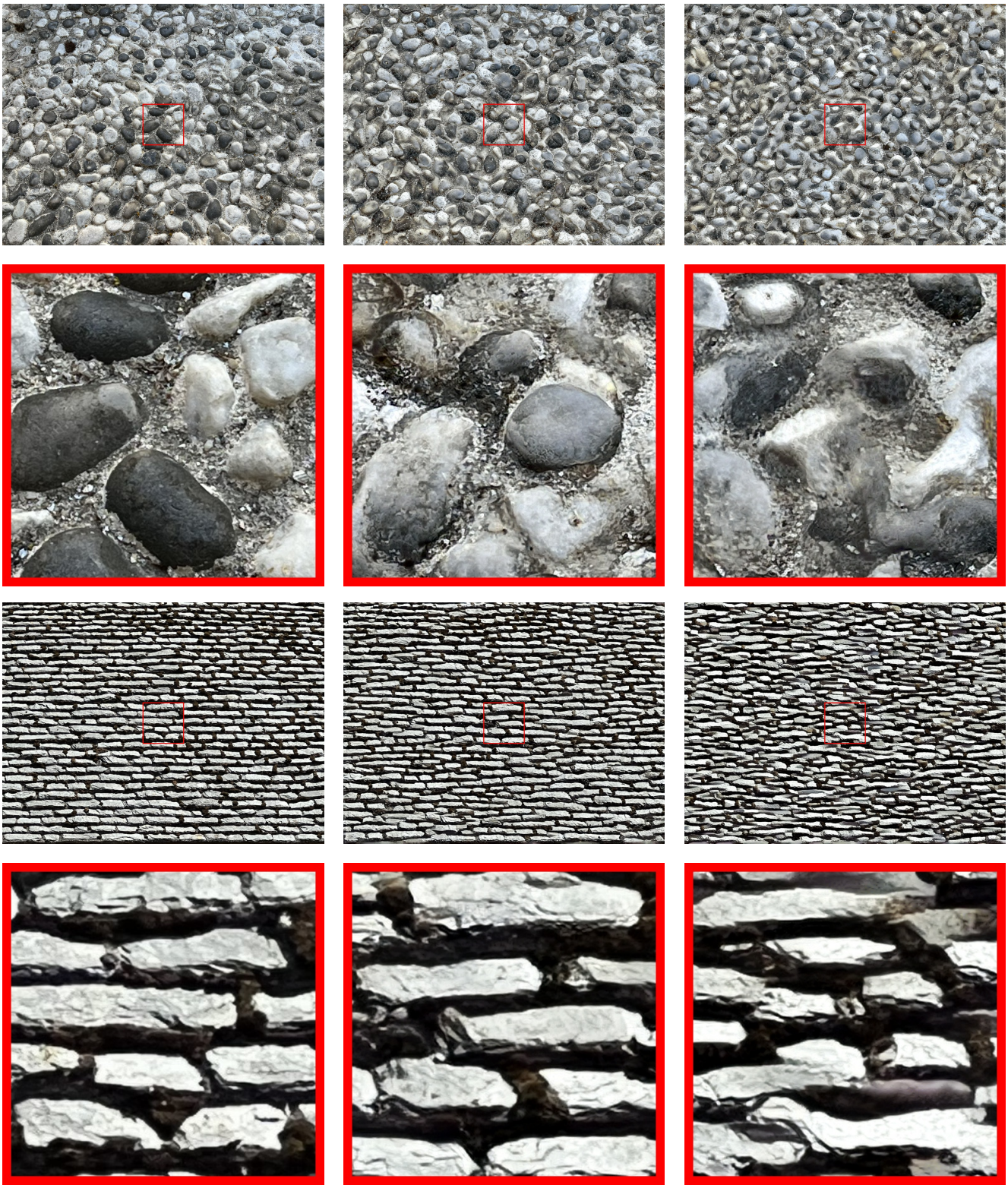

Figure 16. UHR texture synthesis (same as Figure 15): From left to right: Input texture image, synthesis using 5 scales, synthesis using 3 scales. Image have size (3024×4032) (downscaled by a factor 4 for inclusion in the .pdf) and true resolution details have size (512×512)

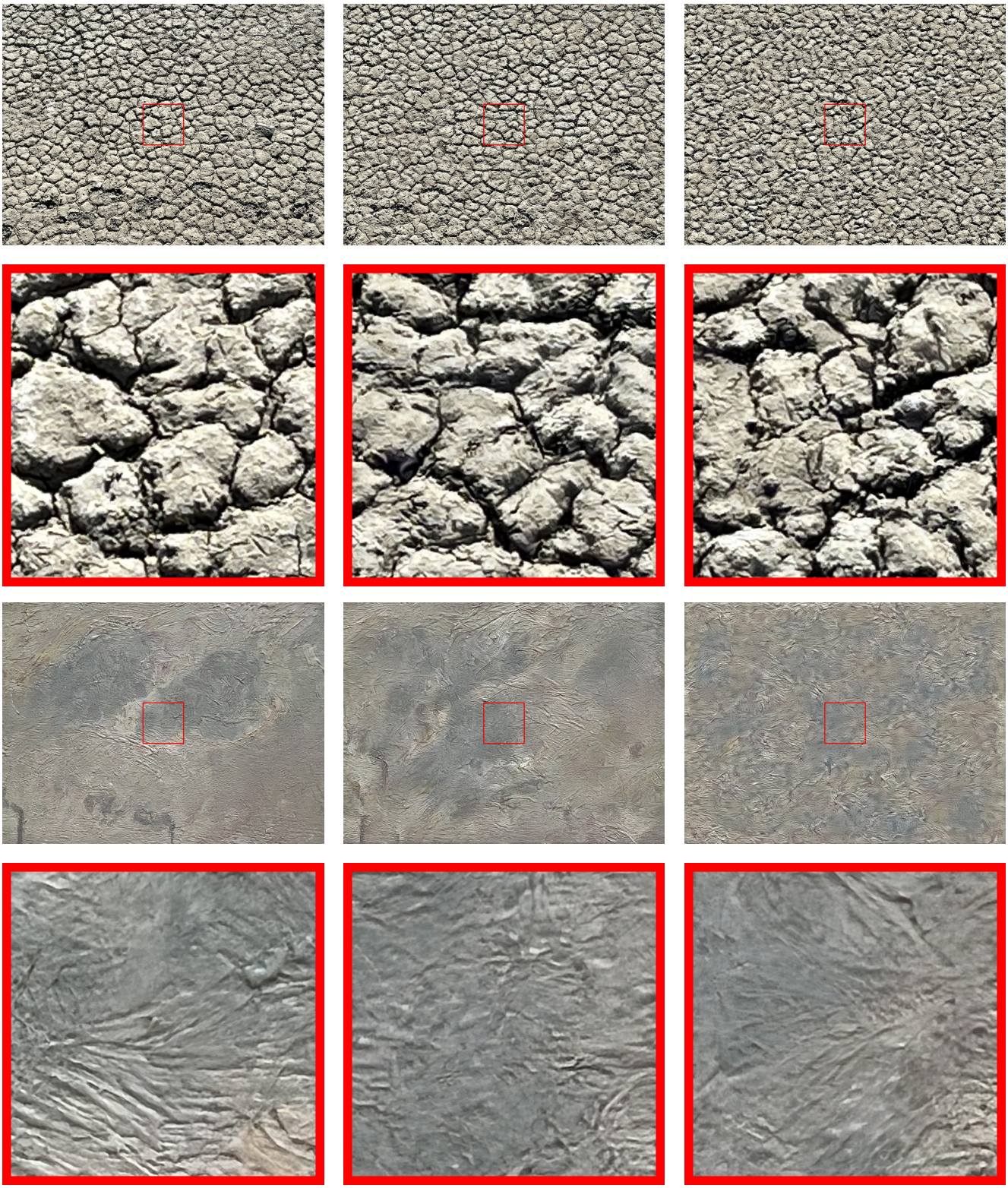

Figure 17. UHR texture synthesis (same as Figure 15): From left to right: Input texture image, synthesis using 5 scales, synthesis using 3 scales. Image have size (3024×4032) (downscaled by a factor 4 for inclusion in the .pdf) and true resolution details have size (512×512)

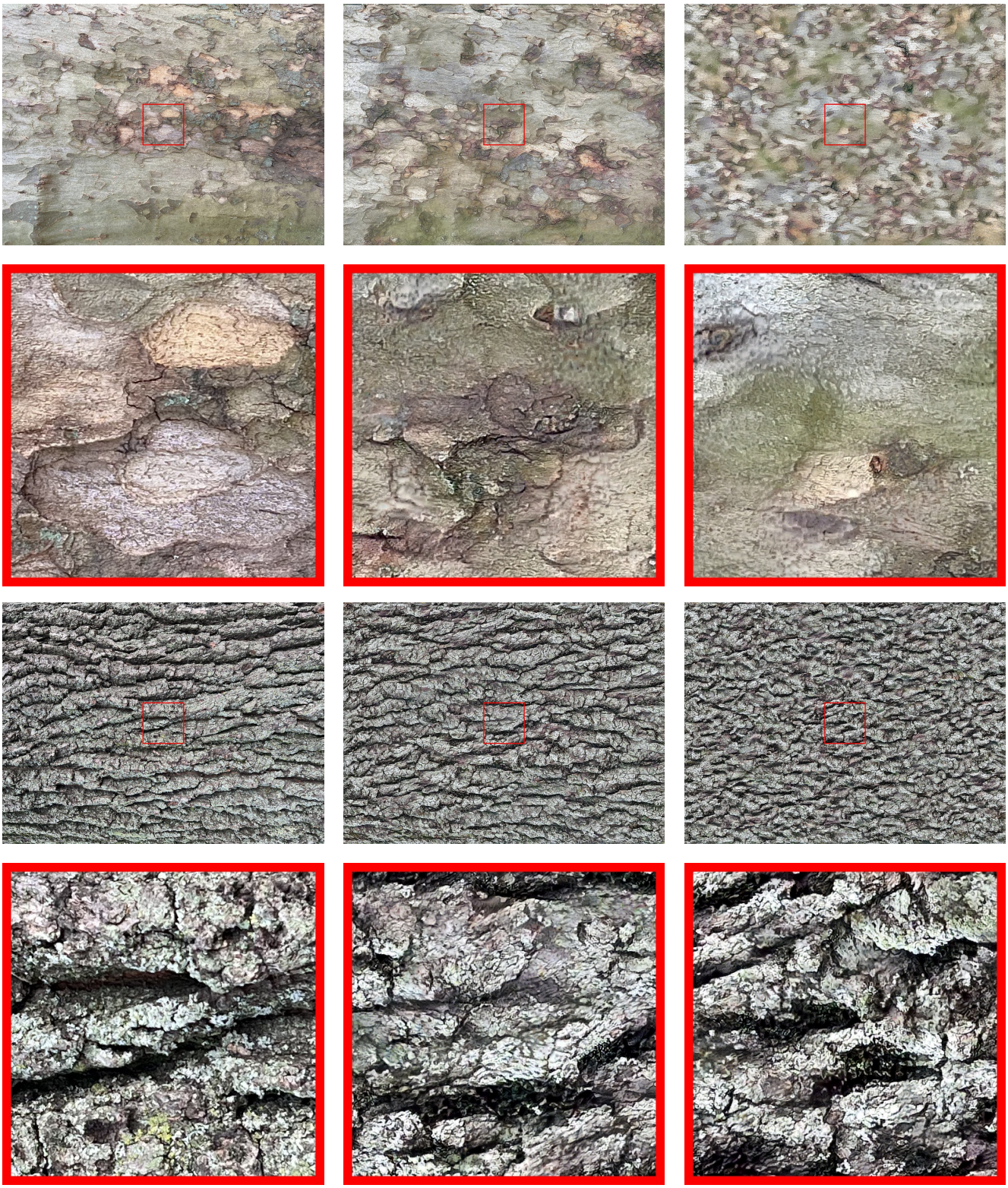

Figure 18. UHR texture synthesis (same as Figure 15): From left to right: Input texture image, synthesis using 5 scales, synthesis using 3 scales. Image have size (3024×4032) (downscaled by a factor 4 for inclusion in the .pdf) and true resolution details have size (512×512)