# SIMuLDiTex: A Single Image Multiscale and Lightweight Diffusion Model for Texture Synthesis

Pierrick Chatillon, Julien Rabin, David Tschumperlé

# SIMuLDiTex: A Single Image Multiscale and Lightweight Diffusion Model for Texture Synthesis

Pierrick Chatillon Julien Rabin David Tschumperlé
Université Caen Normandie, ENSICAEN, CNRS, Normandie Univ, GREYC
UMR 6072, F-14000 Caen, France
firstname.lastname@unicaen.fr

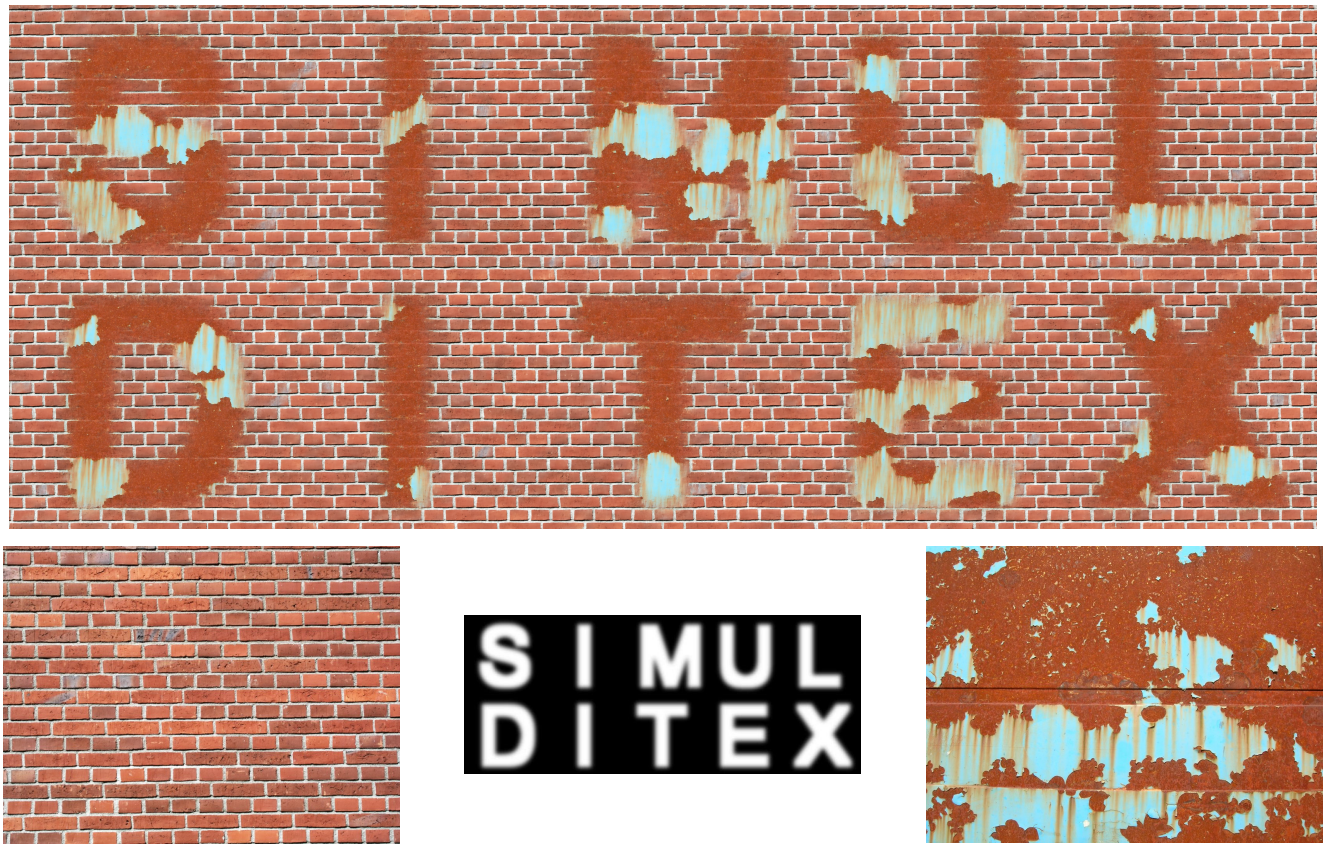



Figure 1. Composite Texture synthesis at 8K resolution (53 s inference for 10.240 × 4.096 px with NVIDIA RTX 4000 ADA 20Go).

## Abstract

*We introduce **SIMuLDiTex**, a Single Image Multi-scale & Light-weight Diffusion Texture model. Recent generative models, though effective, suffer from high computational costs and memory requirements. Our approach addresses these challenges by employing a coarse-to-fine strategy that accelerates sampling and maintains high-resolution fidelity without the need for pre-trained auto-encoders. We introduce a scale-conditioning mechanism that ensures spatial consistency across scales in the diffusion process using a compact U-Net with only one million parameters. Experiments demonstrate that SIMuLDiTex outperforms existing methods in speed, quality, and scalability, offering a practical solution for interactive-time and high-resolution texture generation. Code and models are available at* https://github.com/PierrickCh/SIMuLDiTex.

**Acknowledgments** This work was partially funded by the Normandy Region through the excellence label IArtist project.

## 1. Introduction

**Context** This work addresses the problem of fast and high-quality texture synthesis from a single example. A fundamental challenge in computer vision and graphics, texture synthesis has wide applications in image and video editing. Given a single input image, the goal is to generate a high-resolution texture that preserves the perceptual characteristics and structural consistency of the original while ensuring diversity and realism (see Fig. 1). Traditional methods, such as patch-based approaches [10, 35, 37], utilize low-dimensional parametric models and efficient nearest patch search algorithms (*e.g.*, [2]). These allow for fast synthesis—both in inference and parameter estimation—and can generate arbitrarily large images. However, they struggle with visual fidelity for complex textures and lack generalization. More recently, advanced machine learning methods have emerged, employing large parametric neural networks that overcome these limitations but require more data and sophisticated modeling to avoid overfitting. State-of-the-art image synthesis now combines several key elements: neural representations built upon specific architectures (*e.g.*, auto-encoders, recurrent networks, feed-forward networks) and various modeling techniques, including variational auto-encoders [34], adversarial latent generative models [32], auto-regressive models [49], transformers [11], diffusion models [24], and rectified flow [12]. Additionally, successful conditional generation and editing approaches leverage foundation models that provide sophisticated representations learned from large datasets, such as perceptual features (*e.g.*, LPIPS [73]) or multi-modal aligned representations (CLIP [47]).

**Challenges and limitations** Despite advances, generating arbitrarily large images quickly remains challenging, as state-of-the-art generative models incur high computational costs and memory footprints that, coupled with hardware limitations, restrict their resolution. A key practical challenge with approaches utilizing foundation models is balancing fidelity to the original texture with the diversity these powerful models enable. This contrasts with lightweight models trained on single examples, which struggle to avoid overfitting and realistically extrapolate from limited patterns. The broader field of few-shot image generation [31, 39] further highlights the difficulty of stabilizing sophisticated models like GANs [20] on limited data. Moreover, while techniques exist to compose large synthesis from models with limited receptive fields, preserving long-range correlations remains difficult, especially as computation and memory requirements grow non-linearly with advanced techniques such as attention mechanisms.

**Contributions** This paper introduce **SIMuLDiTex**, a *lightweight* and *multi-scale* diffusion model for *fast* texture synthesis from *a single* image. Our method efficiently captures multi-scale and long-range patterns while maintaining high-quality synthesis of stationary textures at high resolutions. Unlike conventional diffusion models requiring extensive sampling steps, our approach accelerates sampling through a coarse-to-fine strategy while preserving fine-grained details. By working directly in pixel space, we avoid the artifacts and extra parameters of auto-encoders. To ensure spatial consistency across scales during inference, we introduce a novel scale conditioning mechanism depicted in Fig. 2. Our lightweight U-Net architecture contains only one million parameters yet trains efficiently on limited data. We compensate for the absence of attention mechanisms with Fourier modules [6], which capture patterns while enabling arbitrary resolution synthesis with linear complexity in output size. We validate our approach on diverse textures, demonstrating superior performance in speed, quality, and scalability compared to existing methods. As showcased in Figs. 1 and 3, our results highlight the potential of small diffusion models for fast, high-resolution texture generation with minimal computational requirements.

**Outline** The remainder of this paper is organized as follows: Section 2 reviews generative models for texture synthesis; Section 3 describes our proposed model; Section 4 presents experimental analysis, with additional experiments and technical details in the supplementary material; and Section 5 offers concluding discussion.

## 2. Related work

**Texture synthesis with neural features** Stemming from previous methods [22, 44] based on conventional image filters, the groundbreaking work of [16] introduced synthesis from noise by optimizing a statistical criterion representing the exemplar image in a feature space derived from pre-trained deep neural networks (VGG [59]). Numerous approaches have since demonstrated the value of neural features for texture synthesis and editing, including with guidance maps [76]. However, this optimization process remains computationally expensive, requiring a new optimization for each synthesized sample. To address this limitation, [66] proposed shifting from slow pixel optimization to training neural networks that minimize the texture distance defined by [16], enabling rapid sampling from stationary noise. Alternatively, [3, 29, 57, 74] rely on adversarially training generative networks [20] to produce images with the correct patch distribution. SinGAN [57] employs a PatchGAN discriminator examining only patches, as introduced in [28]. [75] combines both texture loss and GAN-based supervision for high-resolution, non-stationary texture synthesis. A significant limitation of both PatchGAN-

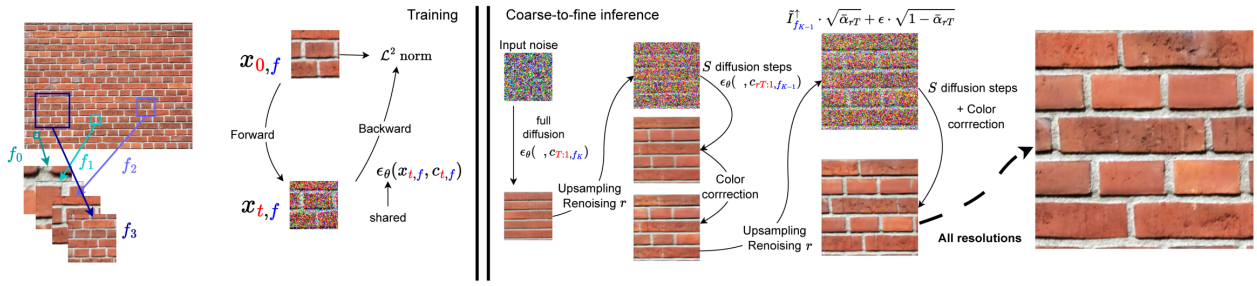



Figure 2. **Overview of SIMuLDiTex.** *Left:* The network is trained as a denoiser, conditioned on the noise level $t$, and the scale factor $f$. *Right:* During inference, the image is synthesized in a coarse-to-fine fashion, each step consisting of a smooth upsampling, the addition of noise of strength corresponding to the diffusion timestep $rT$, a small number of DDIM denoising steps $S$, and a color correction.

Figure 3. **Texture synthesis with SIMuLDiTex**. *Left:* training exemplar image. *Right:* Synthesis with $S = 10$ and $r = 0.8$. Inference takes around 20 seconds. Each image is 10,000 pixels wide but has been highly compressed to meet file size limitations. Please refer to the supplementary material for low-compression samples.

based methods and those minimizing [16]'s texture distance is their local focus, which fails to capture long-range structures (as shown in Fig. 4). To address this, [56] considers spatial correlations of deep features alongside texture distance minimization, improving the synthesis of textures with periodic structures. [19] extends this approach with a coarse-to-fine strategy to better handle long-range dependencies. Further innovations from [3, 4, 38] tackle fast periodic texture synthesis by explicitly incorporating latent sinusoidal waves into generative networks.

**Single image generation** SinGAN [57], designed for image generation from a single training sample, shares several core attributes with classical texture synthesis methods: it employs a multi-scale approach [22, 44, 60], uses coarse-to-fine progression [13, 35], and trains small generative networks at each scale to produce images matching the patch distribution of the input at the corresponding scale [13, 26]. Some researchers have achieved comparable performance by replacing the generative networks with patch-based methods [5, 21], underscoring the connection between single-image synthesis and texture synthesis and highlighting the importance of multi-scale approaches. While SinGAN's use of zero-padding within generative networks effectively captures global object positions (*e.g.* sky above, ground below), this becomes disadvantageous for stationary texture synthesis, as absolute coordinate information disrupts stationarity.

**Auto-regressive models** While adversarial training pio-

neered photo-realistic image synthesis [33], other more stable generative approaches have emerged. Parametric autoregressive models for images parallel patch-based editing techniques like [2], which have proven highly effective for texture synthesis in both non-parametric [9] and parametric models [45]. These models train generative neural networks to iteratively predict sequences of realistic tokens decodable into images, either predicting pixel colors directly, as in pixelCNN [67], or predicting latent representations later decoded into images (VQVAE2 [49]). For infinite tapestry scrolling synthesis, adaptations of such models were proposed in [36]. Drawing inspiration from transformer-based language models [68], transformer-decoders for image generation were quickly explored [42], culminating in state-of-the-art text-to-image models like Dall-E [48], despite requiring massive parameter counts. Latent autoregressive models subsequently reduced model size by an order of magnitude [11]. These latent representations are sufficiently powerful to be edited directly through non-parametric patch sampling without training autoregressive models, as demonstrated for image stylization [37, 53] and few-shot generative modeling [54].

**Diffusion models for texture synthesis** Diffusion models have demonstrated impressive generation capabilities [24], though at the cost of numerous sampling steps during inference. This computational burden can be reduced by trading quality for speed, and the inference process can be made deterministic with a single noise map realization [62]. Following [11], the *Latent Diffusion* model [50] reduces sampling costs by operating in the latent space of a pre-trained auto-encoder [34], where spatial dimensions are reduced. While latent compression degrades image quality, larger models can mitigate this effect [43]. Recent research has focused on accelerating inference through distillation techniques [55].

Generating high-resolution images remains an active research area. Patch-DM [8], similar to LatentPatch [54], creates large images by constructing a latent patchwork. Other works address the computational bottleneck of attention modules in U-Net denoising networks, with [72] proposing a state-space model alternative to capture correlations. Additionally, alternatives to Gaussian noise perturbation have emerged, such as cold diffusion [1].

For texture synthesis specifically, the recent *Infinite Texture* [70] approach fine-tunes a pre-trained text-to-image model. Users first prompt the Latent Diffusion model with a text description to create an example texture. Since this image already exists within the model's generative manifold, fine-tuning restricts the model to create variations of this specific texture. While this fine-tuning is relatively fast using low-rank approximation techniques like LORA [27], inference from such large models remains computationally expensive and restricted to the pre-trained model's resolution. An alternative for editable non-stationary textures appears in [77], introducing a modified backward diffusion with cross-attention to incorporate user guidance.

SinDiffusion [69], inspired by SinGAN, bridges the gap between SinGAN and diffusion models by training a diffusion model on a single image. To prevent overfitting while maintaining sample diversity, they reduce the denoising network's receptive field and draw parallels between diffusion denoising steps and SinGAN stages. However, without multi-scale image decomposition in the spirit of SinGAN, this approach limits long-range correlation reproduction and requires the entire diffusion process at pixel resolution. Works like [25] and [50] demonstrate the effectiveness of low-resolution diffusion followed by dedicated upsampling networks, while [52] employs pyramidal diffusion sampling with a single network across different resolutions.

## 3. Method

In this section, we present our methodology for both training and inference procedures illustrated in Fig. 2 and referred to as SIMuLDiTex. We first detail the multi-scale representation used to represent images in Sec. 3.1 before detailing the design of the lightweight scale-conditional denoising network in Sec. 3.2. The network is trained to learn the backward diffusion process for all the resolutions of the image (Sec. 3.3). Section 3.4 details the inference procedure, a coarse-to-fine approach where coarser resolutions are smoothly upsampled and re-noised before performing a small number of backward diffusion steps. Additionally, the color distribution is adjusted during diffusion to control the synthesis, as explained in Sec. 8.4. Finally, we investigate how our generative model can be adapted to other applications than pure texture synthesis in Sec. 3.5. Practical details are reported in Sec. 8

### 3.1. Multi-scale patch extraction

The goal of this paragraph is to describe the multi-scale representation of the input exemplar used by the model. Let $I$ be the single training image. We denote as $I_f$ the image downscaled by zoom factor $f \leq 1$ for both spatial dimensions, using an interpolation kernel applied on each color channel to perform subsampling. In practice, we rely on the Python library [58] with the linear kernel and anti-aliasing filter. Unless specified otherwise, we use $f \in \mathcal{F} := \{f_k = 1/\sqrt{2}^k; k \in [0...K]\}$ with $K = 6$ to build a pyramid representation $\{I_f, f \in \mathcal{F}\}$ of the original image $I$ at different resolution. For instance $I_1 = I$ and $I_{1/\sqrt{2}^6} = I_{1/8}$ is the image zoomed out by a factor 8.

In order to parallelize training, we take fixed-size square crops (or patches) from this image pyramid $\mathcal{I}$, which can then be arranged in a batch. A sample of random patches for every considered zoom factor in $\mathcal{F}$ is displayed in Fig. 5. We denote by $P$ the linear operator extracting all overlapping patches inside an image $I$. The set of all patches at a

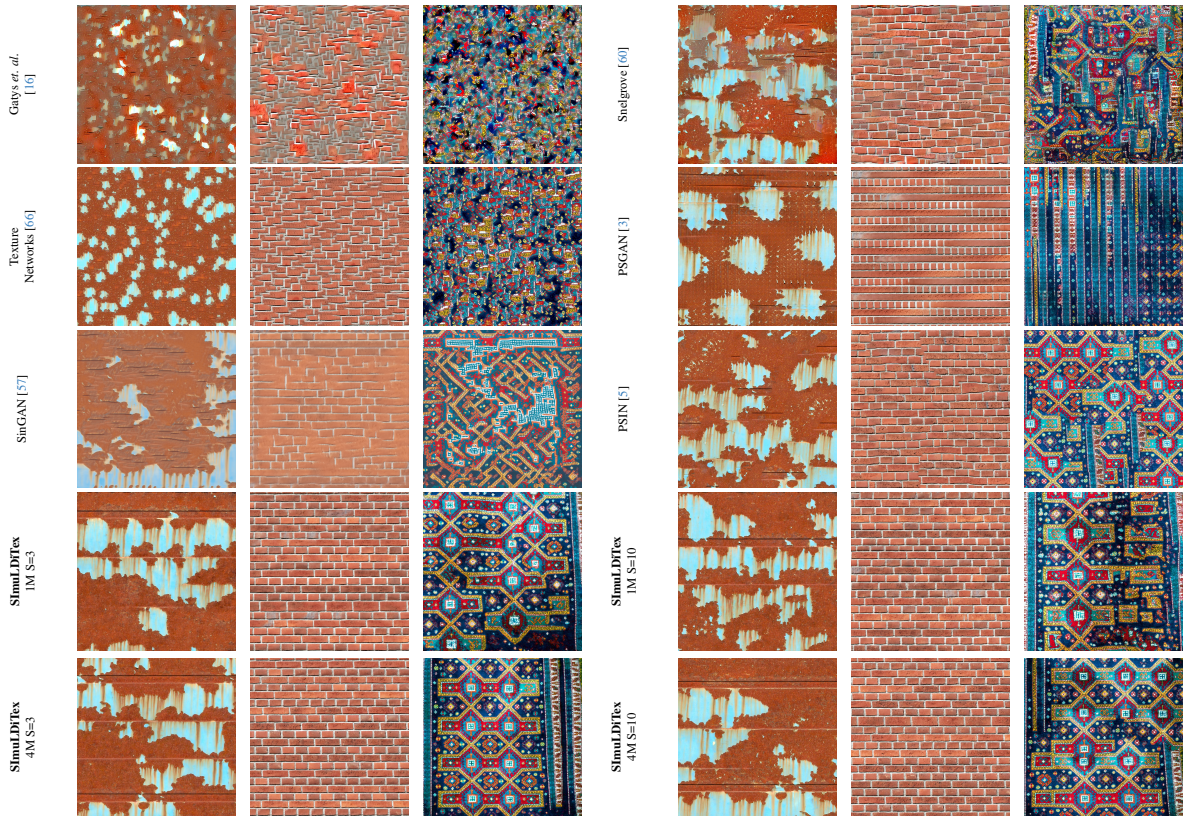



Figure 4. Visual inspection of 2048 by 2048 synthesis for various methods. Training sample images are depicted in Fig. 3.

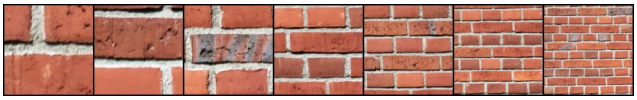


Figure 5. Random patches from $I_f$ for $f$ ranging from 1 to $\frac{1}{8}$.

zoom factor $f$ is referred to as $\mathcal{P}_f := \{p \in P(I_f)\}$. The radius of the patches is fixed to 128 pixels for training.

### 3.2. Denoising network

Our generative model, described in the next section, is built upon a Gaussian denoising diffusion process. During inference (denoising or backward process), random samples are generated iteratively by using a neural network, predicting the additive noise used during the forward process to obfuscate the original data. We follow the popular choice of a U-Net [51] for the denoising network. Note that we remove all the attention layers from the denoising U-Net to significantly reduce its memory footprint.

Taking inspiration from the impressive inpainting result of periodic textures obtained by Lama [64], we residually perform a Fast Fourier Convolution [6] in the bottleneck of the U-Net. This idea has been successfully applied by the authors of Latent Diffusion [50] for the inpainting task. This feature is very important and allows the network to synthesize the structure of periodic textures, as illustrated in Fig. 8. We provide our network with the scale factor $f$ in the same way it is conventionally provided with the diffusion time $t$: First, a sinusoidal embedding $\text{Embd}_i$ yields a representation vector that is then fed to the corresponding multilayer-perceptron $\text{MLP}_i$. The conditioning vector $c$ used to modify the statistics of the features throughout the U-Net (following the method FILM [41]) reads:

$$c_{t,f} = \text{MLP}_1(\text{Embd}_1(t)) + \text{MLP}_2(\text{Embd}_2(f)). \quad (1)$$

An analysis of the difference between the proposed approach and Pyramidal DDPM [52], where absolute positions of pixels are fed to the denoising network, is detailed in the supplementary material (Sec. 8.2).

### 3.3. Diffusion training

Combining notations from the previous paragraph with conventional ones from the literature on diffusion models, we denote from now on by $x_{t,f}$ an image indexed by a scale factor $f$ and a time step $t$. Diffusion models [24, 61] are latent variable models of the form

$$p_\theta(x_{0,f}) := \int p_\theta(x_{0:T,f})\, dx_{1:T,f} \quad (2)$$

where $x_{1:T,f} = (x_{1,f}, \dots, x_{T,f})$ are latents of the same dimensionality as the data $x_{0,f} \sim q_f$. In our framework, the unknown distribution $q_f$ is empirically approximated using patch samples $p \in \mathcal{P}_f$. The joint distribution $p_\theta(x_{0:T,f})$ is called the reverse process, and it is defined as a Markov chain with learned transitions. Starting at a unit normal distribution prior $p_0(x_{T,f}) = \mathcal{N}(x_{T,f}|\mathbf{0}, \mathbf{I})$ for any scale factor $f$, it writes:

$$p_\theta(x_{0:T,f}) := p_0(x_{T,f}) \prod_{t=1}^{T} p_\theta(x_{t-1,f}|x_{t,f}), \quad (3)$$

defining $p_\theta(x_{t-1,f}|x_{t,f})$ as the learned backward process. It is modeled by a Gaussian distribution

$$p_\theta(x_{t,f}|x_{t+1,f}) := \mathcal{N}(x_{t,f}|\mu_\theta(x_{t+1,f}, t, f), \sigma^2_{t,f}\mathbf{I}), \quad (4)$$

where $\mu_\theta$ are the trainable parameters, and the standard deviation $\sigma_{t,f}$ is set to constant for simplicity as in [24].

What distinguishes Gaussian diffusion models from other types of latent variable models is that the approximate posterior $q(x_{1:T,f}|x_{0,f})$, called the forward process or diffusion process, is fixed to a Markov chain that gradually adds Gaussian noise to the data according to a decreasing variance schedule $\beta_{1:T} \in (0, 1]^T$

$$\begin{aligned} q(x_{1:T,f}|x_{0,f}) &:= \textstyle\prod_{t=1}^{T} q(x_{t,f}|x_{t-1,f}) \\ q(x_{t,f}|x_{t-1,f}) &:= \mathcal{N}(x_{t,f}|\sqrt{1-\beta_t}x_{t-1,f}, \beta_t I). \end{aligned} \quad (5)$$

Training is performed by optimizing the usual variational bound on negative log-likelihood, which can be seen as a denoising problem.

As in [24], $\mu_\theta$ in Eq. (4) is parametrized by a denoising deep neural network $\epsilon_\theta$, however conditioned by $c_{t,f}$ defined in (1) in our framework:

$$\mu_\theta(x_{t,f}, t, f) := \frac{1}{\sqrt{1-\beta_t}} \left( x_{t,f} - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta \left( x_{t,f} | c_{t,f} \right) \right) \quad (6)$$

where $\bar{\alpha}_t := \Pi_{\ell=1}^{t} 1 - \beta_\ell$. It is tasked with predicting the noise that was added to a random patch from $I_f$, at the time step $t$ of the forward diffusion process, conditionally to $t$ and $f$. The corresponding loss reads:

$$\mathcal{L}(\theta) = \mathbb{E}_{t,f,p,\varepsilon} \left\| \epsilon_\theta(\sqrt{\bar{\alpha}_t}p + \sqrt{1-\bar{\alpha}_t}\varepsilon \mid c_{t,f}) - \varepsilon \right\|^2 \quad (7)$$

where the expectation is computed over uniform random variables for time $t \in [1..T]$ and scale $f \in \mathcal{F}$ (see Sec. 8.5 for more details), uniform patches $p \in \mathcal{P}_f$ as defined in Sec. 3.1, and unit gaussian noise $\varepsilon \sim \mathcal{N}(\varepsilon|\mathbf{0}, \mathbf{I})$.

### 3.4. Coarse-to-fine sampling

In DDIM [62], authors showed that high-quality samples could be sampled with much fewer steps than during training where typically $T = 1000$. They devise a different sampling sequence using the deterministic noise provided by the denoising network itself. Plugged into our setting at a given scale factor $f$, using $\varepsilon_{t,f} = \epsilon_\theta(x_{t,f}|c_{t,f})$, this yields

$$x_{t-1,f} = \frac{1}{\sqrt{1-\beta_t}} \left( x_{t,f} - \sqrt{1-\bar{\alpha}_t}\varepsilon_{t,f} \right) + \sqrt{1-\bar{\alpha}_{t-1}}\varepsilon_{t,f}. \quad (8)$$

Now, inspired by previous approaches in the literature for texture synthesis, we propose a new coarse-to-fine sampling approach based on the learned diffusion model, as illustrated in Fig. 2. At the coarsest resolution $f_K$, a complete diffusion inference is performed, using our single network conditioned on that coarse scale $\epsilon_\theta(x_{t-1,f_K}|c_{t-1,f_K})$ from $t = T$ to $t = 1$. This preliminary step is fast because the corresponding spatial resolution is limited.

Then, all the resolutions used during training are iteratively considered, from scale factor $f_{K-1}$ to $f_0 = 1$. For each resolution $f_k$, the synthesis of the previous resolution $x_{0,f_{k-1}}$ is smoothly upsampled using bilinear interpolation to define $\tilde{x}_{0,f_k}$. Then, rather than initializing the diffusion process from pure noise $x_{T,f_k}$, we start from an intermediate time $t = rT$ using the forward model Eq. (5) to define $x_{rT,f_k}$ by combining $\tilde{x}_{0,f_k}$ with $x_{T,f_k}$, with the appropriate noise level. The parameter $0 \leq r \leq 1$ is the ratio parameter controlling the noise level, as depicted in Fig. 2. More details are given in the supplementary (see Eq. (10) in Sec. 8.1). Since most of the structure is already present in the image, we only take a small number $S$ of DDIM sampling steps Eq. (8), corresponding to time steps linearly sampled between $s = rT$ and $s = 1$, to update the synthesis of the current scale $f_k$. Unless specified otherwise, we opt experimentally for $S = 3$ as it provides very fast sampling speed. Finally, the color histogram of the image is prescribed to match the exemplar, using sliced Wasserstein distance as detailed in the Sec. 8.4.

### 3.5. Applications

**Large texture synthesis** During inference of very large samples (8192 px), the computation of fine resolutions may not fit within the GPU memory limit. We choose to compose such images from patches that are synthesized iteratively and arranged into a patchwork. Unfortunately, to take into account the boundary effects involved by zero-padding in the convolutional U-Net, some special care is required when combining neighboring patches. For each scale $f$, the proposed solution detailed in the supplementary (Sec. 8.3) consists in averaging overlapping patches, taking into account the receptive field of the denoising network after several steps. The pseudo-algorithm in Tab. 3 summarizes this coarse-to-fine patchwork inference, which allows the synthesis of arbitrarily large images.

**Interpolation** Given two texture images, we train separate models $\epsilon_{\theta_j}$, for $j \in \{0, 1\}$. Each learns to generate samples from $q_j$, the distribution of images with the same texture features as the corresponding exemplar. Given an interpola-

tion parameter $0 \leq \omega \leq 1$, we can generate an image sampled from $q \propto q_0^{1-\omega}\, q_1^{\omega}$ by simply using linear combination of networks $\epsilon_\theta = (1-\omega)\,\epsilon_{\theta_0} + \omega\,\epsilon_{\theta_1}$ in Eq. (8). Allowing $\omega$ to vary across spatial dimensions creates smooth linear interpolation visually, as demonstrated in Figs. 1 and 7.

**Stylization** Rather than relying on guidance maps as in Fig. 1, style transfer consists in using a content image to impose large scale structures [17] while reproducing the local patterns from a style image. The proposed framework SIMuLDiTex can be easily adapted this task as demonstrated in the supplementary (Sec. 6).

# 4. Experiments

## 4.1. Qualitative analysis

The ability of SIMuLDiTex to create very large high-quality textures is demonstrated in Fig. 1 and Fig. 3. Low-compressed images are shown in the supplementary to appreciate the capacity of the model to reproduce even fine details (Figs. 17 to 19). A visual comparison with various methods is proposed in Fig. 4.

**Replication** As a sanity check, we evaluate the model's generalization by ensuring that our framework does not simply replicate portions of the training image, as patch-based methods do (*e.g.* [5, 21]). We assess this by looking at a distinctive pattern from the input, and visualize its nearest neighbors in the synthesis Fig. 6. PatchMatch [2] is used to visually confirm the generalization properties of our method in Sec. 7 (supplementary).

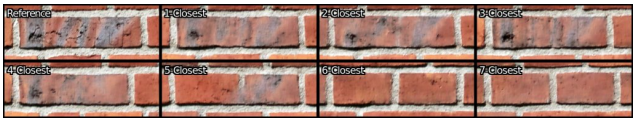



Figure 6. **Nearest-neighbor comparison.** From left to right, top to bottom: reference patch and 1st to 7th closest patch.

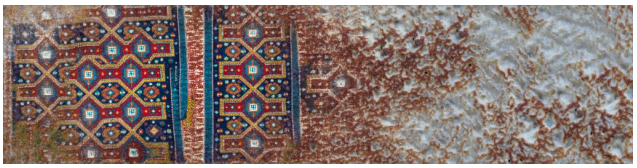


Figure 7. **Spatial interpolation.** Mixing parameter linearly increases from left ($\omega = 0$) to right ($\omega = 1$ for 8,196 px).

**Interpolation** In Fig. 7, the middle part has a wide constant orange area. This color can be found in both the texture exemplars. This corroborates the fact that out generation framework generates patches that are likely to appear in both images. This property may not be always desirable for visually pleasing interpolation between textures.

## 4.2. Quantitative analysis

We compare here quantitatively our approach with various methods from the literature, highlighting its advantage both in terms of image quality and inference/training speed. `SIMuLDiTex 4M` refers to the same model using 4 million parameters instead of 1.

**Time complexity** Tab. 1 provides computation time for different image resolutions using original code if available. For technical and hardware compatibility reasons, note that GPU specifications vary. First, due to memory limitations on GPU, most available code cannot generate images larger than 2028 px. In particular, SinGAN cannot be trained on the images used for the other methods due to memory issues. We crop the image so that no dimension exceeds 1024 before running SinGAN, which notably affects the results.

**Metrics** Tab. 2 compares three conventional metrics to evaluate the synthesis quality with respect to the example texture image: the Gram-loss based on VGG features from [16] (here computed from the 3 first pooling layers, as in [60]), referred to as `Gram-VGG`, the Single Image Fréchet Inception Distance (`SiFID`) [57] based on features from the Inception Network, and the spatial auto-correlation distance. This metric is the mean square error between auto-correlation matrices as used in [56]. The autocorrelation matrices are computed using the Fourier transform (as in AutoCorr [19]). Each metric is computed at 5 resolutions, in order to compare the quality of synthesis of texture patterns of different sizes, the downscaling method being the same as detailed in Sec. 3.1. Note that these metrics do not penalize overfitting as illustrated in [71] for FID [23] from which SiFID [57] is derived from. The metric `Gram-VGG` can vary a lot depending on the input image. To make results more readable, we normalize the score such that the method [60] gets a score of 1. This choice is consistent with the fact that this method optimizes an image to minimize jointly the metric `Gram-VGG` for all considered scale factors.

## 4.3. Ablation

**U-Net bottleneck** In Fig. 8, we compare synthesis obtained with different architectures. Since we work with stationary textures, the attention maps of a multi-head attention module must be stationary as well, not improving the alignment of the synthesis while being expensive for large synthesis. On the other hand, the Fourier Unit in the bottleneck fixes long-range alignment of the pattern.

**Number of scales** In Fig. 9, we investigate the usefulness of the coarse-to-fine approach. Each image is synthesized following the procedure described in Sec. 3, using only the $n$ finest scales. We observe that the quality of details re-

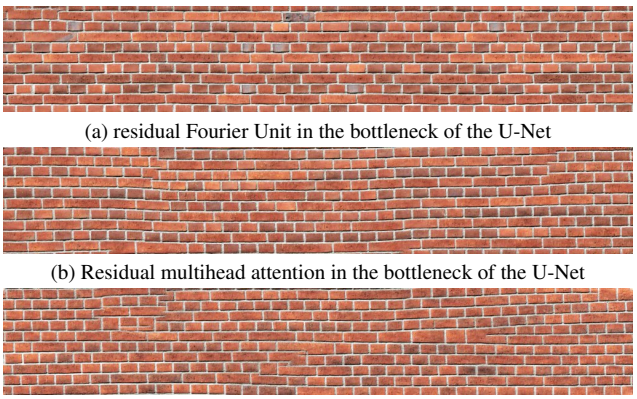
[1] Pytorch implementation by `github.com/JorgeGtz/TextureNets_implementation` for Texture Networks [66].

| Method | Hardware | Parameters (M) | Train time | $256^2$ px | $512^2$ px | $1024^2$ px | $2048^2$ px | $4096^2$ px | $8192^2$ px |
|---|---|---|---|---|---|---|---|---|---|
| Snelgrove et.al. [60] | RTX 4000 ADA | 0 + early VGG | 0 | 10 s | 30 s | 90 s | $\sim$ 2 min | ✗ | ✗ |
| Gatys et.al. [16] | A100 | 0 + VGG | 0 | | | | $\sim$ 10 min | ⋈ | ⋈ |
| PSin [5] | CPU | 0 | 0 | $\sim$ 5 s | $\sim$ 10 s | $\sim$ 1.5 min | $\sim$ 13 min | ⋈ | ⋈ |
| Autocorr [19] | GeForce 1080 Ti | 0 + VGG | 0 | $\sim$ 9 min* | ∅ | $\sim$ 1h* | ✗ | ✗ | ✗ |
| Infinite texture [70] | A100 | $\sim$ 1000 | 10 min* | ∅ | ∅ | ∅ | $\sim$ 6 min* | ∅ | $\sim$ 1h* |
| SinGAN [57] | A100 | 3.3M +3.3M | 6h | | | | 1s | ✗ | ✗ |
| Texture Networks [66][1] | RTX 4000 ADA | 0.5 +VGG | 1h | 30 ms | 32ms | 46 ms | 300 ms | ✗ | ✗ |
| PSGAN [3] | RTX 4000 ADA | 19 (+ 17 Discrr) | 70 min | 0.02 ms | 0.046 ms | 0.2 ms | 0.3 ms | ✗ | ✗ |
| **SIMuLDiTex** S=3 | RTX 4000 ADA | 1.1 | $\sim$ 4h | 0.3s | 0.4 s | 1 s | 3 s | 11 s | 51 s |
| **SIMuLDiTex** S=10 | RTX 4000 ADA | 1.1 | $\sim$ 4h | 0.4 s | 0.6 | 2 s | 7 s | 27 s | 2 min |
| **SIMuLDiTex** 4M S=3 | RTX 4000 ADA | 4.3 | $\sim$ 4h | 1.5s | 1.5 s | 2 s | 5 s | 21 s | 100 s |
| **SIMuLDiTex** 4M S=10 | RTX 4000 ADA | 4.3 | $\sim$ 4h | 1.5 s | 1.8 s s | 3.8 s | 13 s | 1 min | 5 min |

Table 1. **Computation time comparison for training and inference.** * indicates values reported by the original authors (with ∅ for missing resolutions), ✗ denotes a GPU memory error, and ⋈ represents cases where the computation time is excessively long.

| Metric | Time (s) | VGG-Gram (3 layers) ↓ | | | | | SIFID ↓ | | | | | Autocorrelation ↓ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Scale factor | | 1 | $\frac{1}{2}$ | $\frac{1}{4}$ | $\frac{1}{8}$ | $\frac{1}{16}$ | 1 | $\frac{1}{2}$ | $\frac{1}{4}$ | $\frac{1}{8}$ | $\frac{1}{16}$ | 1 | $\frac{1}{2}$ | $\frac{1}{4}$ | $\frac{1}{8}$ | $\frac{1}{16}$ |
| Snelgrove et.al. [60] | 120 | 1 | 1 | 1 | 1 | 1 | 0.09 | 0.03 | 0.05 | 0.09 | 0.17 | 1842.4 | 459.0 | 113.9 | 28.0 | 6.6 |
| Gatys et.al. [16] | 600 | 348.2 | 67.6 | 72.6 | 80.4 | 69.6 | 4.14 | 0.39 | 0.65 | 0.94 | 1.32 | 4225.0 | 1054.5 | 262.8 | 65.1 | 15.8 |
| PSin [5] | 780 | 3.6 | 5.1 | 5.8 | 11.8 | 17.3 | 0.16 | 0.33 | 0.7 | 0.68 | 0.97 | 3568.1 | 889.9 | 221.5 | 54.8 | 13.3 |
| SinGAN [57] | 1 | 337.2 | 318.8 | 292.5 | 181.1 | 63.7 | 1.95 | 1.79 | 1.87 | 1.97 | 2.02 | 4315.5 | 1077.1 | 268.4 | 66.6 | 16.8 |
| Texture Networks [66][1] | 0.3 | **0.8** | **1.2** | 5.0 | 15.4 | 27.0 | **0.06** | **0.08** | 0.25 | 0.87 | 1.48 | 2333.3 | 580.4 | 143.5 | 34.8 | 8.2 |
| PSGAN [3] | 0.0003 | 7.0 | 11.7 | 16.0 | 14.3 | 15.8 | 0.35 | 0.54 | 1.03 | 1.2 | 1.63 | 4874.4 | 1214.4 | 301.5 | 74.0 | 17.7 |
| **SIMuLDiTex** S=3 | 3 | 66.8 | 19.3 | 9.7 | 12.0 | 8.7 | 0.61 | 0.37 | 0.3 | 0.32 | 0.6 | 2202.3 | 547.5 | 135.2 | 32.7 | 7.8 |
| no histogram correction | 3 | 67.3 | 22.3 | 18.2 | 23.0 | 18.5 | 1.03 | 0.78 | 0.72 | 0.65 | 0.85 | 18958.8 | 4735.0 | 1181.9 | 295.0 | 73.4 |
| **SIMuLDiTex** S=10 | 7 | 27.2 | 6.2 | 3.6 | 5.8 | 4.7 | 0.27 | 0.19 | **0.19** | **0.27** | 0.55 | 1585.0 | 394.3 | 97.5 | 23.6 | 5.7 |
| no histogram correction | 7 | 29.0 | 8.6 | 10.5 | 19.5 | 19.0 | 0.52 | 0.48 | 0.56 | 0.53 | 0.75 | 19698.9 | 4919.6 | 1227.8 | 304.8 | 76.3 |
| **SIMuLDiTex** S=50 | 28 | 8.0 | 2.7 | 2.1 | 4.0 | 3.7 | 0.14 | 0.14 | 0.18 | **0.27** | 0.6 | 1807.1 | 449.7 | 111.2 | 27.0 | 6.5 |
| **SIMuLDiTex** 4M S=3 | 5 | 47.0 | 9.8 | 4.5 | 6.5 | 4.3 | 0.47 | 0.24 | 0.22 | **0.26** | **0.5** | **1487.9** | **369.8** | **91.2** | **22.0** | **5.2** |
| **SIMuLDiTex** 4M S=10 | 13 | 14.1 | 2.1 | **1.5** | **3.1** | **3.1** | 0.19 | 0.14 | **0.19** | 0.29 | 0.54 | 1498.7 | 372.2 | 91.7 | 22.1 | **5.2** |

Table 2. **Quantitative comparison for texture fidelity.** Image are synthesized at $2048 \times 2028$ px resolution. See the text for details.



(a) residual Fourier Unit in the bottleneck of the U-Net

(b) Residual multihead attention in the bottleneck of the U-Net

(c) no operation in the bottleneck of the U-Net

Figure 8. **Ablation on U-Net architecture** showing the advantage of the Fourier module (8a), vs with (8b) or without (8b) attention.

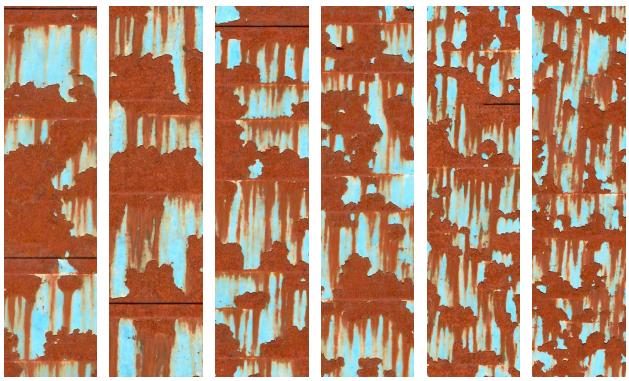
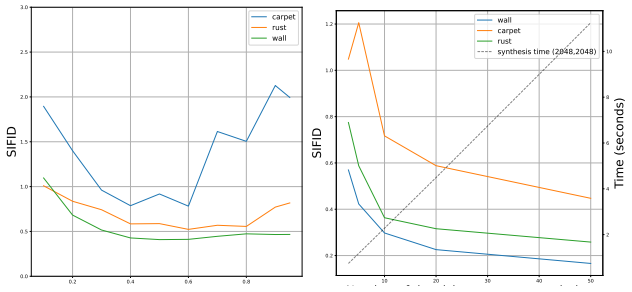



Figure 9. **Abblation on $K$.** Synthesis using a restricted number of resolution factor $f_0..f_K$ in the coarse-to-fine synthesis ($S = 3$).

mains, while the structure is severely damaged as we drop more and more coarser scales.

**Number of timesteps** In Fig. 10, we study the hyper-parameters $r$ and $S$, controlling respectively the amount of noise re-introduced at each scale and the number of denoising steps. In Fig. 10a, the best quality is reached around $r = 0.5$ for most textures, trading-off between adding detail relevant to the current scale ($r = 1$) and keeping coarser structures ($r = 0$). Fig. 10b showcases the trade-off between quality and sampling time with parameter $S$.




(a) Renoising factor $r$ (b) Number of steps $S$ per resolution

Figure 10. **Abblation on $r$ and $S$.** Hyper-parameters selection using SiFID criterion [57] for the coarse-to-fine synthesis.

## 5. Conclusion

We introduced SIMuLDiTex, a lightweight and multi-scale diffusion model designed for fast and high-quality texture synthesis from a single image. By leveraging a coarse-to-fine generation strategy and a novel scale-conditioning mechanism, our approach ensures spatial consistency while significantly accelerating inference compared to other deep learning approaches. With only one million parameters, our compact U-Net architecture efficiently captures multi-scale patterns and enables interacting-time, high-resolution synthesis. Experimental results demonstrate that SIMuLDiTex outperforms existing methods in terms of speed, quality, and scalability, making it a practical solution for texture generation with minimal computational requirements. Future work could explore extensions to non-stationary textures and integration with controllable generation frameworks.

## References


[1] Arpit Bansal, Eitan Borgnia, Hong-Min Chu, and Jie et al. Li. Cold diffusion: Inverting arbitrary image transforms without noise. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:41259–41282, 2023. 4

[2] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009. 2, 4, 7, 3

[3] Urs Bergmann, Nikolay Jetchev, and Roland Vollgraf. Learning texture manifolds with the periodic spatial GAN. In *ICML*, pages 469–477. PMLR, 2017. 2, 3, 5, 8

[4] Pierrick Chatillon, Yann Gousseau, and Sidonie Lefebvre. A geometrically aware auto-encoder for multi-texture synthesis. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 263–275, 2023. 3

[5] Nicolas Cherel, Andrés Almansa, Yann Gousseau, and Alasdair Newson. A patch-based algorithm for diverse and high fidelity single image generation. In *ICIP*, pages 3221–3225, 2022. 3, 5, 7, 8, 4

[6] Lu Chi, Borui Jiang, and Yadong Mu. Fast fourier convolution. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4479–4488. Curran Associates, Inc., 2020. 2, 5

[7] Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *ICLR*, 2023. 1

[8] Zheng Ding, Mengqi Zhang, Jiajun Wu, and Zhuowen Tu. Patched denoising diffusion models for high-resolution image synthesis. In *ICLR*, 2024. 4

[9] A.A. Efros and T.K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 1033–1038, 1999. 4

[10] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346, 2001. 2

[11] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12873–12883, 2021. 2, 4

[12] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024. 2

[13] Bruno Galerne, Arthur Leclaire, and Julien Rabin. A texture synthesis model based on semi-discrete optimal transport in patch space. *SIAM Journal on Imaging Sciences*, 11(4):2456–2493, 2018. 3

[14] Bruno Galerne, Lara Raad, José Lezama, and Jean-Michel Morel. Scaling painting style transfer. In *Computer Graphics Forum*, page e15155. Wiley Online Library, 2024. 1

[15] LA Gatys, AS Ecker, and M Bethge. Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks (2015). *CoRR abs/1505.07376*. 1

[16] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, page 262–270, 2015. 2, 3, 5, 7, 8

[17] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neu-

ral networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 7, 1

[18] Leon A Gatys, Alexander S Ecker, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Controlling perceptual factors in neural style transfer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3985–3993, 2017. 1

[19] 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. 3, 7, 8

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2672–2680, 2014. 2

[21] Niv Granot, Ben Feinstein, Assaf Shocher, Shai Bagon, and Michal Irani. Drop the gan: In defense of patches nearest neighbors as single image generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13460–13469, 2022. 3, 7, 4

[22] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, page 229–238, New York, NY, USA, 1995. Association for Computing Machinery. 2, 3

[23] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. 7

[24] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 2, 4, 5, 6

[25] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23(47): 1–33, 2022. 4

[26] Antoine Houdard, Arthur Leclaire, Nicolas Papadakis, and Julien Rabin. A generative model for texture synthesis based on optimal transport between feature distributions. *Journal of Mathematical Imaging and Vision*, 65(1):4–28, 2023. 3

[27] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022. 4

[28] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017. 2

[29] Nikolay Jetchev, Urs Bergmann, and Roland Vollgraf. Texture synthesis with spatial generative adversarial networks, 2016. 2

[30] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pages 694–711, Cham, 2016. Springer International Publishing. 1

[31] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:12104–12114, 2020. 2

[32] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8107–8116, 2020. 2

[33] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 43(12):4217–4228, 2021. 4

[34] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proc. Conference ICLR*, 2014. 2, 4

[35] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. In *ACM SIGGRAPH 2005 Papers*, pages 795–802. 2005. 2, 3

[36] Jian Liang, Chenfei Wu, Xiaowei Hu, Zhe Gan, Jianfeng Wang, Lijuan Wang, Zicheng Liu, Yuejian Fang, and Nan Duan. Nuwa-infinity: Autoregressive over autoregressive generation for infinite visual synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 15420–15432. Curran Associates, Inc., 2022. 4

[37] Lin Liang, Ce Liu, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics (ToG)*, 20(3):127–150, 2001. 2, 4

[38] Jue Lin, Gaurav Sharma, and Thrasyvoulos N. Pappas. Toward universal texture synthesis by combining texton broadcasting with noise injection in styleGAN-2. *e-Prime - Advances in Electrical Engineering, Electronics and Energy*, 3:100092, 2023. 3

[39] Bingchen Liu, Yizhe Zhu, Kunpeng Song, and Ahmed Elgammal. Towards faster and stabilized gan training for high-fidelity few-shot image synthesis. In *iclr*, 2021. 2

[40] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *ICLR*, 2022. 1

[41] Gabriel Meseguer-Brocal and Geoffroy Peeters. Conditioned-u-net: Introducing a control mechanism in the u-net for multiple source separations. In *20th International Society for Music Information Retrieval Conference*, 2019. 5

[42] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. pages 4055–4064. PMLR, 2018. 4

[43] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. In *The Twelfth International Conference on Learning Representations*. 4

[44] Javier Portilla and Eero P Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International journal of computer vision*, 40(1):49–70, 2000. 2, 3

[45] Lara Raad, Agnès Desolneux, and Jean-Michel Morel. A conditional multiscale locally gaussian texture synthesis algorithm. *Journal of Mathematical Imaging and Vision*, 56:260 – 279, 2016. 4

[46] Julien Rabin, Julie Delon, and Yann Gousseau. Regularization of transportation maps for color and contrast transfer. In *2010 IEEE International Conference on Image Processing*, pages 1933–1936, 2010. 5

[47] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763. PmLR, 2021. 2

[48] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. pages 8821–8831. Pmlr, 2021. 4

[49] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019. 2, 4

[50] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022. 4, 5

[51] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing. 5

[52] Dohoon Ryu and Jong-Chul Ye. Pyramidal denoising diffusion probabilistic models. *ArXiv*, abs/2208.01864, 2022. 4, 5

[53] Benjamin Samuth, David Tschumperlé, and Julien Rabin. A patch-based approach for artistic style transfer via constrained multi-scale image matching. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 3490–3494. IEEE, 2022. 4

[54] Benjamin Samuth, Julien Rabin, David Tschumperlé, and Frédéric Jurie. A non-parametric latent model for face generation. *Available at SSRN 4772632*, 2024. 4

[55] Axel Sauer, Frederic Boesel, Tim Dockhorn, Andreas Blattmann, Patrick Esser, and Robin Rombach. Fast high-resolution image synthesis with latent adversarial diffusion distillation. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024. 4

[56] Omry Sendik and Daniel Cohen-Or. Deep correlations for texture synthesis. *ACM Transactions on Graphics (ToG)*, 36(5):1–15, 2017. 3, 7

[57] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *ICCV*, pages 4570–4580, 2019. 2, 3, 5, 7, 8, 9

[58] Assaf Shocher. Resizeright. `https://github.com/assafshocher/ResizeRight`, 2018. 4, 1

[59] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 2, 5

[60] Xavier Snelgrove. High-resolution multi-scale neural texture synthesis. In *SIGGRAPH Asia 2017 Technical Briefs*, pages 1–4. 2017. 3, 5, 7, 8

[61] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015. 5

[62] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *ArXiv*, abs/2010.02502, 2020. 4, 6

[63] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021. 1

[64] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha,

Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2149–2159, 2022. 5

[65] David Tschumperlé, Sébastien Fourey, and Garry Osgood. G'mic: An open-source self-extending framework for image processing. *Journal of Open Source Software*, 10(105):6618, 2025. 4

[66] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, page 1349–1357. JMLR.org, 2016. 2, 5, 7, 8

[67] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in Neural Information Processing Systems (NeurIPS)*, 29, 2016. 4

[68] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017. 4

[69] Weilun Wang, Jianmin Bao, Wengang Zhou, Dongdong Chen, Dong Chen, Lu Yuan, and Houqiang Li. Sindiffusion: Learning a diffusion model from a single natural image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025. 4

[70] Yifan Wang, Aleksander Holynski, Brian L Curless, and Steven M Seitz. Infinite texture: Text-guided high resolution diffusion texture synthesis. *arXiv preprint arXiv:2405.08210*, 2024. 4, 8

[71] Ryan Webster, Julien Rabin, Loic Simon, and Frédéric Jurie. Detecting overfitting of deep generative networks via latent recovery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11273–11282, 2019. 7

[72] Jing Nathan Yan, Jiatao Gu, and Alexander M. Rush. Diffusion models without attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8239–8249, 2024. 4

[73] Richard Zhang, Phillip Isola, and et al. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, 2018. 2, 1

[74] Xilong Zhou, Milos Hasan, and et al. Tilegen: Tileable, controllable material generation and capture. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022. 2, 5

[75] Yang Zhou, Zhen Zhu, Xiang Bai, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Non-stationary texture synthesis by adversarial expansion. *ACM Trans. Graph.*, 37(4), 2018. 2

[76] Yang Zhou, Kaijian Chen, Rongjun Xiao, and Hui Huang. Neural texture synthesis with guided correspondence. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18095–18104, 2023. 2

[77] Yang Zhou, Rongjun Xiao, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Generating non-stationary textures using self-rectification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7767–7776, 2024. 4

# SIMuLDiTex: A Single Image Multiscale and Lightweight Diffusion Model for Texture Synthesis

## Supplementary Material

This appendix provides additional information and results about the proposed approach and is organized as follows:

- A variant of the proposed approach for image stylization is first described in Sec. 6;
- Additional quantitative and qualitative results for texture synthesis are then displayed in Sec. 7;
- Last, Section 8 provides various additional details:
  - coarse-to-fine sampling (Sec. 8.1);
  - analysis about scale embedding (Sec. 8.2);
  - patch blending (Sec. 8.3);
  - color correction (Sec. 8.4);
  - technical details (Sec. 8.5).

## 6. Image stylization

**Previous work** Another unprecedented key aspect of the seminal work of [15] is the design of a perceptual loss function [17], as coined and built-upon in numerous work later on [30, 73]. It allows to compose a synthetic image blending photo-realistically texture-based characteristics (style) with semantic-based characteristics (content) from two different images. Due to the memory footprint of the back-propagation through VGG, synthesizing high-resolution images yet require time-consuming coarse-to-fine optimization [18]. Recent works address this specific challenge to achieve satisfying style transfer on high-resolution images (see *e.g.* [14]), albeit with a consequent computation time.

**Stylization with SImuLDiTex** Contrarily to SDEdit [40], which achieve Image to Image edition with a score-based model applied to latent diffusion generative model, we do not start the stylization with a noised version of the input. Instead, we apply a tunable fidelity term at each resolution of our multi-scale synthesis, as described below:

Given a low-resolution content image $y$, we perform stylization by generating a high-resolution image following both the texture prior learned by $\epsilon_\theta$, and a fidelity prior $p_{\text{fidel}}$. This new prior $p_{\text{fidel}}$ should aim at assigning high probability to high-resolution images which, after downsampling, are close to $y$. In the following, we measure this similarity in the $L^2$ sense.

Like [7], we combine steps of diffusion denoising, which can be seen as gradient steps maximizing the log-likelihood of the data [63], and steps of fidelity to the input. Using a Gaussian prior and writing $\downarrow$ as the downsampling operator (and $\uparrow$ its adjoint) we have

$$p_{\text{fidel}}(x_{t,f}|y) \propto \exp\left(-\tfrac{1}{\sigma^2}\|y - \downarrow x_{t,f}\|^2\right)$$

which yields the gradient expression w.r.t the first variable

$$\nabla_x \log(p_{\text{fidel}}(x_{t,f}|y)) = \tfrac{1}{\sigma^2}\uparrow(y - \downarrow x_{t,f}),$$

that is straightforward to compute (again, we rely on the auto-differentiable modules from Python library [58]). In practice, the parameter $\zeta = \frac{1}{\sigma^2}$ controls the force of the fidelity prior. Large values of $\zeta$ impose the local means of $x_t$ to match the pixel values of $y$. As a result, values of $\zeta$ higher than 1 overshoot the target, so that in practice we impose $\zeta \in [0, 1]$. An illustration for high-resolution stylization is proposed in Fig. 11 for $\zeta = 0.1$. The proposed approach directly benefits from the mutli-scale synthesis algorithm that is detailed in Sec. 8.

## 7. Additional results

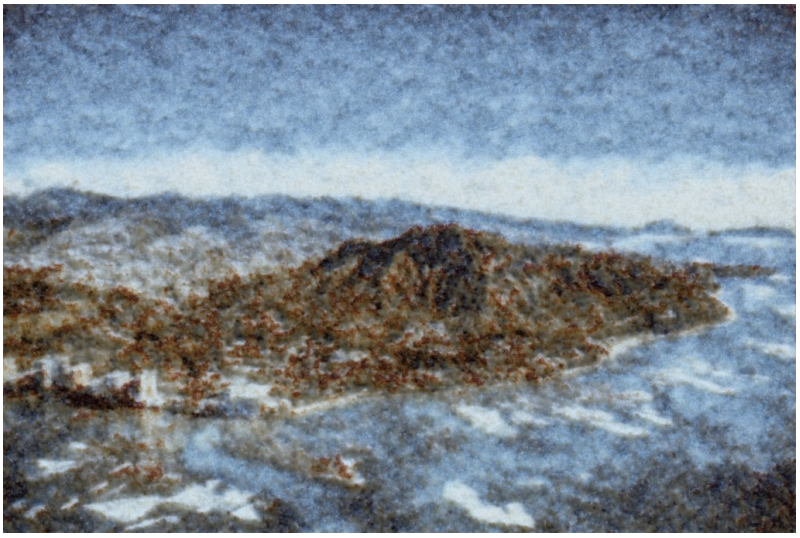
**Visual results** The full resolution version of Fig. 3, is included in Fig. 13. Additional high-resolution texture syntheses are provided in Figs. 17 to 19. These $5700\times4096$ images are less compressed than those in the main article, allowing readers to better assess the high-quality details generated by SIMuLDiTex using $S = 10$, $r = .5$. Fig. 19 uses the 4M version of the model. The original training image is copied on the top left as a reference.

Fig. 12 shows additional results on texture interpolation. Note that in the experiment shown in the article in Fig. 7, the two original textures have completely different color distributions. Yet, without any need for color correction, the proposed interpolation method succeeds in creating a visually appealing result. Nevertheless, some artifacts are noticeable on the border of the image. This is due to the zero-padding involved in the convolutional U-Net.

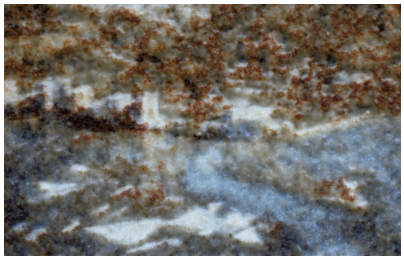
**Visual diversity analysis** The goal of the following experiments is to assess the diversity of the learned generative model. To that end, we investigate two properties: i) the model must not learn some simple pattern that is repeated over and over (variability); and ii) the proposed method should not boil down to verbatim copy exact pattern from the exemplar, as in non-parametric model (generalization). To visually assess these two properties, we use local comparison by extracting patches from images, as already done in Fig. 6, now at a large scale.

(a) Content image.

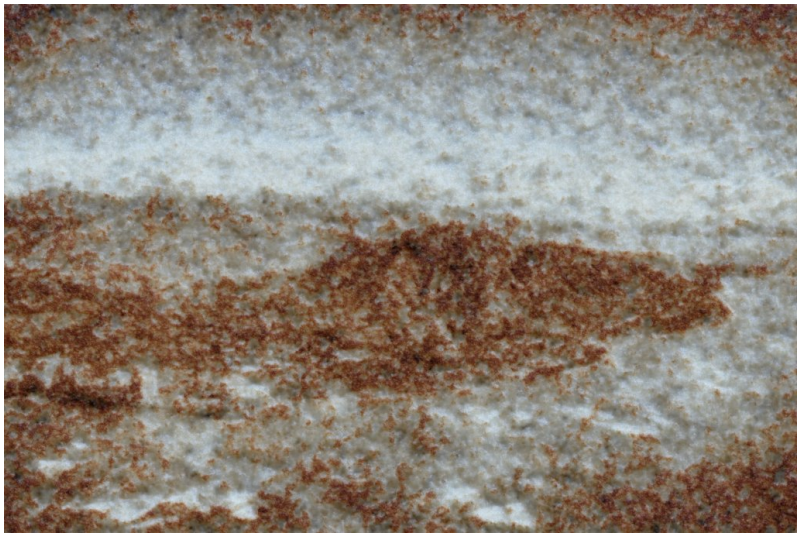
(b) Texture image.



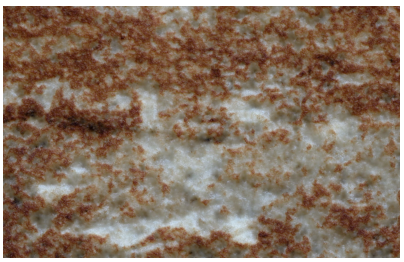
(c) Stylization.



(d) close up.



(e) Stylization after color correction.



(f) close up.

Figure 11. Image stylization using a variant of SIMuLDiTex. The synthesized images (bottom row) are four times larger than the content image. Please zoom in to view high-quality details.

Figure 12. Example of spatial interpolation, given 2 trained models.

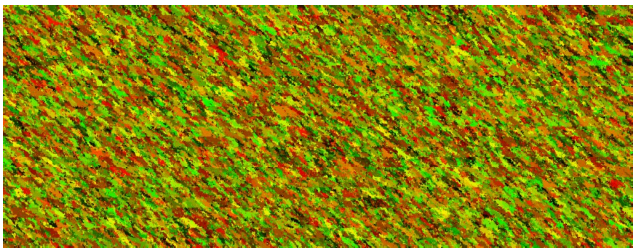
Figure 13. **Texture synthesis with SIMuLDiTex**. Full resolution version of Fig. 3

Since a brute-force comparison of all overlapping patch pairs is computationally prohibitive for high-resolution images, we instead use the PatchMatch algorithm [2] with $17 \times 17$ patches. PatchMatch is an ultra-fast approximate nearest-neighbor search with a strong regularity hypothesis on the correspondence map (or warp) between two images. It is useful to detect if some local regions of an image are copied into another one. Note that, due to the regularity prior of PatchMatch, correspondence maps between two random noise generate random correspondence maps with some local regularity (as shown in Fig. 14b).

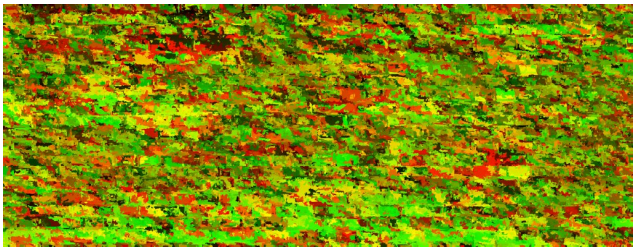
In Fig. 14, we first visually assess whether our method generates sufficiently diverse random samples. For this test, the $10,000$-pixel-wide synthesized 'wall' texture from Fig. 13 (Fig. 3 in the main article) is split into two $5,000$-pixel-wide images. The colormap used for visualization, shown in Fig. 14a, is defined by normalizing pixel coordinates of both images. As a result, the warped colormap visually indicates the position of the approximate nearest exemplar patch in the other image. For an identity warp (Fig. 14a), the colormap indicates that the two images are perfect copies. The warp map for our synthesis, shown in Fig. 14c, demonstrates that no local patterns are exactly replicated.

In Fig. 15, we perform the same experiment to assess the generalization property of the proposed model. This time, PatchMatch algotihm is used between the reference image

(a) Colormap of patch coordinates.



(b) Baseline: warped colormap of the nearest neighbor patch between two random noise images. The small piecewise affine regions observed are a result of the PatchMatch algorithm's regularity prior.



(c) Warped colormap of the nearest neighbor patch between two generated samples.

Figure 14. Copy detection using PatchMatch algorithm, between two random noise (Fig. 14b), and from SIMuLDiTex (Fig. 14c), illustrating the diversity of the generated samples.

used for training and a $10,000$ pixel-wide sample ('wall' textures both displayed in Fig. 3). As a baseline, we compare our approach to a multi-scale patch-based copy method based on the PatchMatch algorithm, similar to [5, 21]. The striking difference demonstrates the generalization capability of our method.

# 8. Additional details

## 8.1. Coarse-to-fine sampling

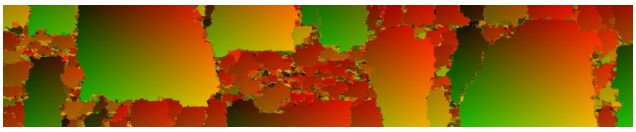
The algorithm in Tab. 3 summarizes the proposed approach to synthesize texture with arbitrary dimensions.

Recall that one of the major advantages of the forward Gaussian diffusion model Eq. (5) used during training is the ability to sample directly noise at any arbitrary time step using:

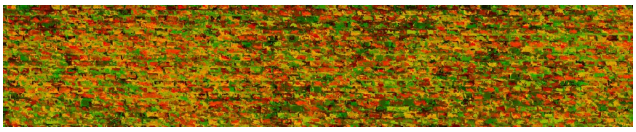
$$q(x_{t,f}|x_{0,f}) := \mathcal{N}(x_{t,f}|\sqrt{\bar{\alpha}_t}x_{t-1,f}, 1-\bar{\alpha}_t I). \tag{9}$$

In our coarse-to-fine inference procedure, this is taken advantage of to blend the upsampled image $\tilde{x}_{0,f_k}$ from the previous scale with a unit-random noise $x_{T,f_k}$ using

$$x_{rT,f_k} := \sqrt{\bar{\alpha}_{rT}}\tilde{x}_{0,f_k} + \sqrt{1-\bar{\alpha}_{rT}}x_{T,f_k}. \tag{10}$$



(a) (baseline) Warp map for a synthesis obtained with multi-scale patch-based method (using the implementation of [65]).



(b) Warp map for our synthesis (wall image in Fig. 3)

Figure 15. **Similarity detection.** Same experiment as in Fig. 14, but now between a reference image and a synthetic texture image. The reference image here is the 'wall' picture from Fig. 3. The warped colormaps are computed using samples from two generative models: a patch-based copy technique (top) and the proposed method SIMuLDiTex (bottom). Regular areas in such correspondence maps suggest local copies from the input texture.

## 8.2. Positional embedding vs scale conditioning

In this paragraph, we compare our network conditioning strategy with the approach from Pyramidal diffusion [52]. The authors provide absolute coordinates to the diffusion network through positional embedding. Positional embeddings are sine maps of the absolute pixel coordinates, with varying frequencies. Following their notations, let $\boldsymbol{i} : x \longrightarrow x$ be the identity mapping, for a pixel coordinate $1 \leq x \leq N$ at the coarsest scale. They use an upsampled version of $\boldsymbol{i}$ for synthesis at higher resolutions: $U_f(\boldsymbol{i}) : x \longrightarrow \frac{x}{f}$, for $1 \leq x \leq fN$. First, we want to demonstrate that with such positional information, the network is implicitly provided with the scale factor $f$. Indeed, $f$ can be recovered by the network as the inverse of the amplitude of the gradient operator on the positional embedding. Let PE be a positional embedding sine wave of fixed frequency $\xi$ noted as $\text{PE}(x) = \sin(\xi \cdot U_f(\boldsymbol{i})(x))$. Then:

$$\begin{aligned} \nabla\text{PE}(x) &= \nabla sin(\xi \cdot U_f(\boldsymbol{i}(x))) \\ &= \frac{\xi}{f} \cdot cos(\xi \cdot \frac{\boldsymbol{i}(x)}{f}). \end{aligned} \tag{11}$$

Using scale conditioning upon training a pyramidal model is crucial, as shown by the authors of [52]. While positional embeddings work well for aligned datasets like FFHQ, they should be avoided for texture synthesis, which requires positional variability. To ensure this, the denoising network must not receive absolute position information. Instead, we condition it directly on the scale factor $f$, just as it is conditioned on the non-local diffusion step $t$.

**Input:** Output Image dimensions, diffusion network $\epsilon_\theta$, color distribution
**Hyperparameters:** number of diffusion steps $T$, noise ratio $r$, number of sampling steps $S$, patch size, scale factors $\mathcal{F} = \{f_1, \dots f_K\}$
**Output:** Synthesized image $x_{0,f_0}$

▷ *Step 1: Initialization at coarse resolution $f_K$*

**Step 1.1:** Sample noise $x_{T,f_K} \sim \mathcal{N}(x_{T,f_K}|\mathbf{0}, \mathbf{I})$ with the specified dimensions downscaled at the coarse resolution $f_K$
**Step 1.2:** Sample overlapping patches $p_{T,f_K}$ from $x_{T,f_K}$ (see Sec. 8.3)
**Step 1.3:** Perform a full DDIM diffusion inference at the coarsest resolution $f_K$ from $t = T$ to $t = 1$ using (8) on each patch $p_{t,f_K}$
**Step 1.4:** Set $x_{0,f_K}$ by blending patches $p_{0,f_K}$
**Step 1.5:** Apply color correction (see Sec. 8.4)

▷ *Step 2: Iterative refinement at finer resolutions*

**for** $k = K - 1$ **to** $0$ **do**
  **Step 2.1:** Set $\tilde{x}_{0,f_k}$ by upsampling $x_{0,f_{k+1}}$ to resolution $f_k$
  **Step 2.2:** Sample noise $x_{T,f_k} \sim \mathcal{N}(x_{T,f_k}|\mathbf{0}, \mathbf{I})$ at the same dimensions
  **Step 2.3:** Set $x_{rT,f_k}$ from $\tilde{x}_{0,f_k}$ and $x_{T,f_k}$ using Eq. (10)
  **Step 2.4:** Sample overlapping patches $p_{rT,f_k}$ from $x_{rT,f_k}$ (see Sec. 8.3)
  **Step 2.5:** Perform partial DDIM sampling with $S$ steps from $s = rT$ to $s = 1$ using Eq. (8) on each patch $p_{s,f_k}$
  **Step 2.6:** Set $x_{0,f_k}$ by blending patches $p_{0,f_k}$
  **Step 2.7:** Apply color correction (see Sec. 8.4)
**end for**
**Return** $x_{0,f_0}$

Table 3. SIMuLDiTex Coarse-to-Fine diffusion patchwork for texture synthesis of arbitrary dimensions.

### 8.3. Patch blending

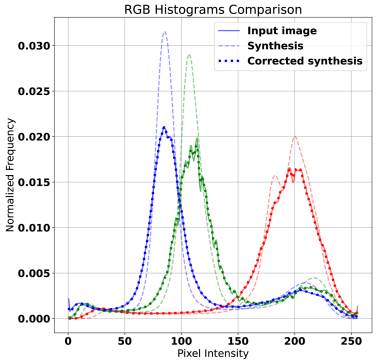
**Boundary effect for large images** As previously mentioned, the use of zero-padding in generative models such as SinGAN [57] creates undesirable artifacts for texture synthesis. Similar boundary effects occur in synthesis based on convolutional neural networks (CNNs). The VGG-19 CNN [59] used for feature extraction in various texture synthesis methods employs zero-padding, which similarly affects synthesis results. Additionally, zero-padding compromises the translation equivariance of CNNs, except when circular convolutions are used in generative networks, as in [66, 74]. Consequently, while CNNs can function as feed-forward generative models for synthesizing large images beyond GPU memory capacity using a patchwork approach, careful stitching is required to address boundary effects caused by padding.

**Proposed solution** When creating an image of arbitrary large dimensions, the proposed coarse-to-fine patch sampling comes in handy. However, due to the boundary conditions with the U-Net denoising network (relying on convolutions with zero-padding), overlapping patches sampled from the same random initialisation image $x_{T,f_K}$ may not coincide exactly on the overlapping area. While the difference is not important, it may become noticeable and create seams when composing a large image as a patchwork. Taking into account the size of the receptive field of the U-Net (about 51 pixels), we simply synthesize overlapping patches that are blended by averaging colors to compose the intermediate image $\tilde{I}_f$. We found this simple procedure efficient to discard any perceptually visible artifacts.

### 8.4. Color correction

During inference, we apply a post-processing step to the image $\tilde{I}_f$, the output of our multi-scale diffusion sampling method at resolution $f$. Using the Sliced-Wassertein 2 distance, we prescribe the color distribution of $\tilde{I}_f$ to match the color distribution of $I_f$. To do so, we take 20 steps of histogram prescription (as described in [46]) along randomly sampled unit color vectors $v$. We verify that the color distribution indeed matches that of the exemplar image in Fig. 16a, and observe the effect of this correction on a crop of a generated image in Fig. 16b. For this illustration, the color correction was only applied as a post-processing step. The color correction steps re-establishes the correct contrast, highlighting the details generated. The need for

this correction can be explained in two ways. First, the small size of the network may explain its generative limitations. Secondly, the same denoising network is trained on many resolutions, and downsampled images tend to be smoother than their counterparts of finer resolution. As a consequence, the network is biased towards smoother images. An ablation study on the application of this correction is reported in Tab. 2.




(a) RGB histograms of the exemplar image (top), a 10000 × 1920 synthesis (middle), with color matching (bottom).

(b) Without (top) and with (bottom) color correction.

Figure 16. Impact of the color correction as post-processing treatment.

## 8.5. Technical details

**Implementation** The code to define our diffusion model has been adapted from the Pytorch implementation of https://github.com/lucidrains/denoising-diffusion-pytorch. The metrics in Tab. 2 are computed on 5 different synthesis for the 3 images (wall, carpet and rust).

**Training** In the loss function Eq. (7), the expectation is theoretically computed from uniformly distributed random time variable $t \in \{1, ..T\}$ and scale factors $f$ in the discrete set $\mathcal{F}$. The number of time steps is set to $T = 200$ during training, using patches of 128 pixels randomly sampled from the image pyramid representation. However, our inference devotes most of the time steps ($T = 200$) on the coarsest resolution $f_K$. Indeed, the remaining diffusion steps requires $S = 3$ steps for each other scale factors $f_{K-1}..f_0$), that is a total of 18 steps for $K = 6$. As a result, during training, the variable $f$ in Eq. (7) is sampled with probability $\frac{1}{4}$ for $f_K$, and the rest uniformly on all other scales (*i.e.* with probability $\frac{1}{12}$).
For all experiments, the optimizer is Adam with default parameters, using a learning rate of $10^{-4}$ and $100k$ iterations. The batch size is 32.

Figure 17. Low-compressed synthesis example. Synthesis takes 38 s.

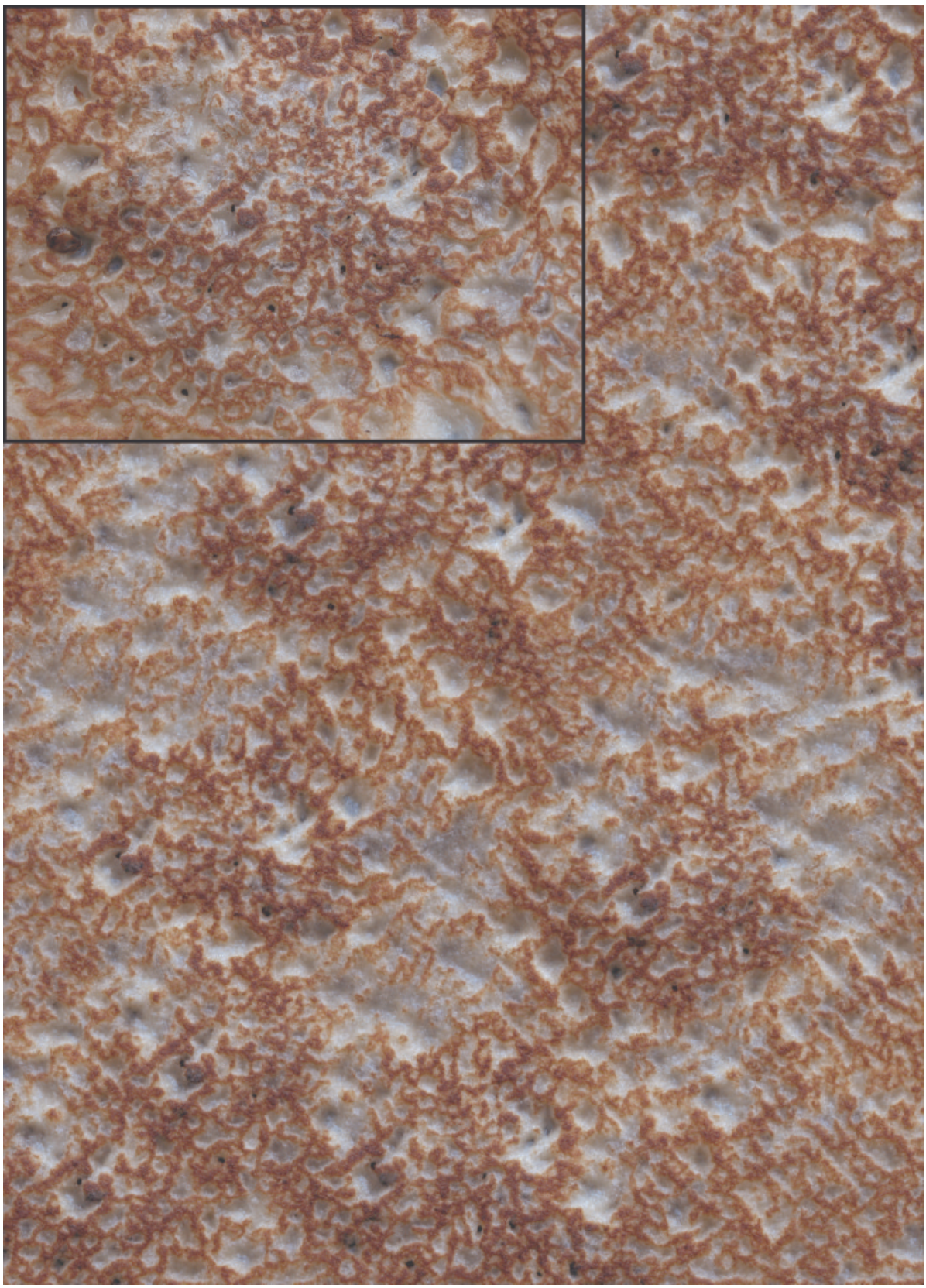


Figure 18. Low-compressed synthesis example. Synthesis takes 38 s.

Figure 19. Low-compressed synthesis example. Synthesis takes 77 s.