# Wasserstein Generative Models for Patch-based Texture Synthesis

Antoine Houdard, Arthur Leclaire, Nicolas Papadakis, Julien Rabin

# Wasserstein Generative Models for Patch-based Texture Synthesis

**Antoine Houdard**
Univ. Bordeaux
CNRS, IMB
UMR 5251, France

**Arthur Leclaire**
Univ. Bordeaux
CNRS, IMB
UMR 5251, France

**Nicolas Papadakis**
Univ. Bordeaux
CNRS, IMB
UMR 5251, France

**Julien Rabin**
Normandie Univ., UniCaen
ENSICAEN, CNRS
GREYC, UMR 6072
France

## Abstract

In this paper, we propose a framework to train a generative model for texture image synthesis from a single example. To do so, we exploit the local representation of images via the space of patches, that is, square sub-images of fixed size (e.g. $4 \times 4$). Our main contribution is to consider optimal transport to enforce the multiscale patch distribution of generated images, which leads to two different formulations. First, a pixel-based optimization method is proposed, relying on discrete optimal transport. We show that it is related to a well-known texture optimization framework based on iterated patch nearest-neighbor projections, while avoiding some of its shortcomings. Second, in a semi-discrete setting, we exploit the differential properties of Wasserstein distances to learn a fully convolutional network for texture generation. Once estimated, this network produces realistic and arbitrarily large texture samples in real time. The two formulations result in non-convex concave problems that can be optimized efficiently with convergence properties and improved stability compared to adversarial approaches, without relying on any regularization. By directly dealing with the patch distribution of synthesized images, we also overcome limitations of state-of-the art techniques, such as patch aggregation issues that usually lead to low frequency artifacts (e.g. blurring) in traditional patch-based approaches, or statistical inconsistencies (e.g. color or patterns) in learning approaches.

## 1 Introduction

Image synthesis consists in creating photorealistic pictures while prescribing some desired attributes. Two main strategies have been investigated in the literature. One can either sample an image distribution using a *generative model* learnt from a large image dataset, as in GANs [10]. On the other hand, one can turn to *exemplar-based synthesis*, that is, generating new images which exhibit features that are similar to the ones of a *single example*, as in [8, 27] (possibly using "perceptual features" extracted with a neural network trained on an image database [15]). The latter is the main topic of this paper. In the literature of exemplar-based synthesis, most effort has focused on texture synthesis, which can produce a large texture sample from a small observation, in an efficient manner. The exemplar texture is often assumed to be perceptually stationary, *i.e.* with no large geometric deformations nor changes in lighting transformations, and we adopt this stationary setting here. Texture models can be broadly classified between parametric [33, 25, 8] and non-parametric [5, 20] models. Related applications include dynamic texture synthesis [30], texture inpainting [6], 3D texture mapping [13], texture interpolation [31], morphing [2], expansion [32] or procedural generation [14].

**Motivation** In this work, we consider the local representation of images obtained by the extraction of patches, which are small sub-images of size $s \times s$ (where $s$ usually ranges from 3 to 16). This

Preprint. Under review.

representation takes profit of the self-similarities of natural images, and subsequently lies at the core of very efficient image restoration methods [3, 34, 21]. It is particularly well adapted to textural content where structural redundancies can be exploited to form new textures with a simple iterative copy/paste procedure [5]. Besides, considering global statistics on patches was proven fruitful in designing efficient and stable texture models [7]. In the following, we use optimal transport (OT) distances to compare patch empirical distributions in a relevant way.

While deep learning approaches have recently shown impressive performance for image synthesis [16], patch-based methods are still competitive when only a single image is available for training [2, 27], both considering the computational cost and the visual performance. Moreover, deep learning methods are still difficult to interpret, whereas patch-based models offer a better understanding of the synthesis process and its cases of success and failure. However, patch-based approaches suffer from three main limitations in practice. To begin with, patches are often processed independently and then combined to form a recomposed image. This leads to low frequency artifacts such as blurring because the patches overlap [20]. In addition, optimization has to be performed sequentially in a coarse-to-fine manner (both in image resolution and patch size) starting from a good initial guess. Last, global patch statistics must be controlled along the optimization to prevent strong visual artifacts [17]. We tackle all these aspects in the proposed OT framework.

**Related work** The model proposed in this paper falls into the scope of texture optimization, which formulates synthesis as the minimization of an energy (that may encode visual features or global statistics) starting from a random initialization. Such a framework can embrace famous parametric texture synthesis algorithms through a pixelwise optimization, *e.g.* [25] (which matches responses to a bank of complex steerable filters) and [8] (which matches responses to a pre-learnt neural network). A related framework has been introduced in [20], with an energy that reflects, at multiple scales, the distances of patches from the synthesis to the ones of the exemplar. The corresponding synthesis algorithm consists in iterating patch nearest-neighbor projections in a coarse-to-fine manner. The main drawback of this energy is that it is oblivious of the global statistics of the output image, and thus exhibits trivial minima. This model has been improved in [12] where discrete OT plans are iteratively used to enforce the patch distribution of the exemplar. Because of the cost of discrete OT, this algorithm has strong limitations in terms of computational time and output size. In contrast, the models of [7, 22] are based on semi-discrete OT maps which can be estimated offline, and thus copes with these constraints. In parallel, several models based on Generative Adversarial Networks [10] have been proposed, which allow for feed-forward synthesis of general images [27] or texture images [29, 2] from a single image example. When considering the training of a generative network on an image dataset based on the minimization of the discrepancy between distributions, the latter introduction of Wasserstein GANs [1] (WGAN) has offered an elegant solution to mode collapse issues.

Alternative techniques to train generative convolutional neural network (CNN) also took profit from the OT framework (*e.g.* Sliced-Wasserstein distance in the latent space of auto-encoder [19]). Although achieving state-of-the art performance, GANs suffer from some limitations. To begin with, GANs require to optimize a discriminative network, which makes the process unstable and requires a large number of additional parameters [10, 24]. However the dual formulation of Wasserstein-1 distances allows to restrict to 1-Lipschitz discriminative networks. Different strategies has thus been proposed to enforce such a constraint (*e.g.* weight clipping or gradient penalty [11]), thus only approximating the true Wasserstein-1 distance. In [4], the optimisation of the Wasserstein distance in WGAN is driven by the semi-discrete formulation of OT between the discrete distribution of training images and the density of generated images. Finally, we note that the OT approach in Generative Networks is mainly considered for comparing distributions of generated images and not for prescribing statistics on a single synthesized image.

**Contributions** In this context, we propose to use OT to constrain the patch distribution of the synthesized image to be close to the one of an unique example image. By proposing a formulation that directly handles the patches of the generated image, our work implicitly addresses the aforementioned limitations of patch-based methods (aggregation, multi-scale optimization, and statistical control). The main contributions are (i) A new and versatile framework for image synthesis from a single example using multi-scale patch statistics and optimal transport; (ii) The computation of the gradient in the discrete and semi-discrete settings used to derive stable algorithms for image synthesis and

generative network training from a single example; (iii) Application of the proposed framework for image optimization and generative network for texture synthesis and inpainting.

**Outline** The paper is organized as follows. To begin with, we recall the OT framework in Section 2. In Section 3, we propose a new model for image synthesis *via* explicit optimization of the discrete formulation of the Wasserstein distance between patch distributions, at different resolutions. Then, we consider in Section 4 the case where the synthesized images are generated by a convolutional neural network trained with a stochastic algorithm using the semi-discrete Wasserstein formulation. Additional experiments (including comparisons with state of the art methods and analysis of the results) as well as technical details are provided in the supplementary material.

## 2 Background on Optimal Transport

Let $\mathcal{X}, \mathcal{Y}$ be two compact spaces included in $\mathbf{R}^d$ and $c : \mathcal{X} \times \mathcal{Y} \to \mathbf{R}$ be a continuous cost function. We consider two measures $\mu, \nu$ supported on $\mathcal{X}, \mathcal{Y}$, respectively, and we denote by $\Pi(\mu, \nu)$ the probability distributions on $\mathcal{X} \times \mathcal{Y}$ having marginals $\mu$ and $\nu$.

**Definition 1** (OT cost and Wasserstein distance)**.** *The OT cost is defined by*

$$\mathrm{OT}_c(\mu, \nu) = \inf_{\pi \in \Pi(\mu,\nu)} \int c(x, y) d\pi(x, y). \tag{1}$$

*If $c(x, y) = \|x - y\|^p$, $p \geq 1$, then $W_p(\mu, \nu) = \mathrm{OT}_c(\mu, \nu)^{\frac{1}{p}}$ defines the $p$-Wasserstein distance.*

In the following, theoretical results will be formulated for a general OT cost but the experiments focus on the $W_2$ cost. Also, we will repeatedly use the dual formulation of OT.

**Theorem 1** (Semi-dual formulation [26])**.** *If $\mathcal{X}$ and $\mathcal{Y}$ are compact and the cost $c$ is continuous, then*

$$\mathrm{OT}_c(\mu, \nu) = \max_{\varphi \in \mathcal{C}(\mathcal{Y})} \int \varphi^c(x) d\mu(x) + \int \varphi(y) d\nu(y), \tag{2}$$

*where $\varphi : \mathcal{Y} \to \mathbf{R}$ and its $c$-transform is defined by $\varphi^c(x) = \min_{y \in \mathcal{Y}} [c(x, y) - \varphi(y)]$.*

## 3 Image optimization

In this section we formulate an energy that will constrain the patch distribution of an image to be close to a target distribution for the OT cost. Using the semi-dual formulation of OT, we end up with a non-convex concave saddle-point problem. We then propose a pixelwise optimization algorithm for this energy, which exhibits a good empirical behavior (stability and convergence). Finally we extend this energy in a multiscale fashion in order to address texture synthesis, and we make the connection with the texture optimization framework of [20].

### 3.1 Setup of the problem

For an image $u \in \mathbf{R}^n$ with $n$ pixels, we consider the collection of its patches $Pu = (P_1 u, \ldots, P_n u)$, that is, the list of all sub-images of size $s \times s$ extracted from $u$. To simplify, we consider periodic boundary conditions so that the number of patches is exactly $n$. Notice that $P_j$ is a linear operator whose adjoint $P_j^T$ maps a given patch $q$ to an image whose $j$-patch is $q$ and is zero elsewhere. Consider also the empirical patch distribution of an image $u \in \mathbf{R}^n$ defined by $\mu_u = \frac{1}{n} \sum_{i=1}^n \delta_{P_i u}$. Given an example texture image $v \in \mathbf{R}^m$, we aim at generating an image $u \in \mathbf{R}^n$ whose patch distribution $\mu_u$ is close to $\mu_v$ for the OT cost.

From the semi-dual formulation of Theorem 1, this amounts to minimize the function

$$w(u) = \mathrm{OT}_c(\mu_u, \mu_v) = \max_{\varphi \in \mathbf{R}^m} f(\varphi, u), \quad \text{where} \quad f(\varphi, u) = \frac{1}{n} \sum_{i=1}^n \varphi^c(P_i u) + \frac{1}{m} \sum_{j=1}^m \varphi_j. \tag{3}$$

Solving (3) is now equivalent to

$$\min_{u \in \mathbf{R}^n} \max_{\varphi \in \mathbf{R}^m} f(\varphi, u). \tag{4}$$

From the OT theory [26], we know that $f$ is concave in $\varphi$ which will be helpful in practice. However, the function $f$ is not convex in the second variable $u$, and thus we will focus on approaching local minima of $w$ in (3).

### 3.2 Theoretical results

In this paragraph, we will study the differential properties of the function $f$ introduced in (3). The first result, stated without proof, is a standard derivation of the gradient w.r.t. $\varphi$.

**Theorem 2.** *Let $u \in \mathbf{R}^n$ and $\varphi \in \mathbf{R}^m$. For any $i$, let $j^\star(i) \in \arg\min_j c(P_i u, y_j) - \varphi_j$ (with an arbitrary choice in case of* ex-aequo*). Then, $f(\cdot, u)$ admits a super-gradient at $\varphi$ given by*

$$g_j(\varphi, u) = \frac{1}{m} - \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{j^\star(i)=j} \tag{5}$$

The regularity in $u$ is essentially linked to the differential property of the Wasserstein distance w.r.t. its arguments, which is here linked to the positions of points in $\mu_u$. To analyze the situation, we consider the open Laguerre cells $L_j(\varphi) = \{x \mid \forall k \neq j,\ c(x, y_j) - \varphi_j < c(x, y_k) - \varphi_k\}$. For any $\sigma : \{1, \dots, n\} \to \{1, \dots, m\}$, denote $\mathcal{A}_\sigma(\varphi) = \{u \in \mathbf{R}^n \mid \forall i, P_i u \in L_{\sigma(i)}(\varphi)\}$ and $\mathcal{A}(\varphi) = \sqcup_\sigma \mathcal{A}_\sigma(\varphi)$. In other words, $\mathcal{A}(\varphi)$ is the set of images whose patches have no *ex-aequo* values for $\varphi^c$. Since $c$ is continuous, it is straightforward to see that $\mathcal{A}_\sigma(\varphi)$ is an open subset of $\mathbf{R}^n$.

**Theorem 3.** *Assume that $c$ is differentiable w.r.t. the first variable. Let $\varphi \in \mathbf{R}^m$ and $u \in \mathcal{A}_\sigma(\varphi)$. Then $f(\varphi, \cdot)$ is differentiable at $u$ and*

$$\nabla_u f(\varphi, u) = \frac{1}{n} \sum_{i=1}^{n} P_i^T \nabla_u c(P_i u, y_{\sigma(i)}). \tag{6}$$

*Proof.* Since $\mathcal{A}_\sigma(\varphi)$ is open, for $v$ sufficiently close to $u$, we have also that $v \in \mathcal{A}_\sigma(\varphi)$ and thus

$$f(\varphi, v) = \frac{1}{n} \sum_{i=1}^{n} \big(c(P_i v, y_{\sigma(i)}) - \varphi_{\sigma(i)}\big) + \frac{1}{m} \sum_{j=1}^{m} \varphi_j \tag{7}$$

and the result follows by applying the chain-rule to the first term. □

Finally, the following theorem provides the gradient of $w$ at particular points $u$ where the associated optimal dual potential $\varphi^*$ leads to no *ex-aequo*.

**Theorem 4.** *Let $u \in \mathbf{R}^n$ and $\varphi^\star \in \arg\max_\varphi f(\varphi, u)$ such that $u \in \mathcal{A}(\varphi^\star)$. Then $w$ in* (3) *is differentiable at $u$ and $\nabla w(u) = \nabla_u f(\varphi^\star, u)$.*

*Proof.* Since $u \in \mathcal{A}(\varphi^\star)$ there exists a map $\sigma : \{1, \dots, n\} \to \{1, \dots, m\}$ such that $u \in \mathcal{A}_\sigma(\varphi^\star)$. Since $\varphi^\star$ is optimal and its $c$-transform makes no *ex-aequo* for the patches $P_i u$, we have in particular that the map $P_i u \mapsto y_{\sigma(i)}$ realizes the OT from $\mu_u$ to $\nu$. Then, for $v$ close to $u$, we still have $v \in \mathcal{A}_\sigma(\varphi^\star)$ and thus the same map also realizes the OT from $\mu_v$ to $\nu$, which implies that $\varphi^\star \in \arg\max_\varphi f(\varphi, v)$. Therefore for $v$ close to $u$, we have $w(v) = f(\varphi^\star, v)$ which suffices to conclude. □

Let us emphasize that the hypothesis of Theorem 4 can be true only in very specific cases. Indeed, the fact that $P_i u \mapsto y_{\sigma(i)}$ is an OT map from $\mu_u$ to $\nu$ implies, from mass conservation, that $m$ divides $n$. Fortunately for the applications at hand, such condition is easily met as one can sample the target patch distribution accordingly. In addition, we only use the gradient $\nabla_u f(\varphi, u)$ during optimization.

### 3.3 A one-scale texture synthesis algorithm

We now detail our texture synthesis algorithm that estimates an image $u$ minimizing (3), so that the patch distribution of $u$ is close to the one of an example image $v$. To do so, we look for a local saddle point of the problem (4) with an iterative alternate scheme on $\varphi$ and $u$, starting with an initial noise image $u^0$. For a fixed $u^k$, we perform a gradient ascent, using the super-gradient given in Theorem 2 to obtain an approximation $\varphi^{k+1}$ of $\varphi^* \in \arg\max_\varphi f(\varphi, u^k)$. A gradient-descent step is then realized on $u$, using the gradient given in Theorem 3. In practice we consider the Adam optimizer

[18] with learning rate 0.01. For texture synthesis, in all the experiments, we consider the quadratic cost $c(x, y) = \frac{1}{2}\|x - y\|^2$. In this particular case, the gradient w.r.t. $u$ reads

$$\nabla_u f(\varphi^{k+1}, u^k) = \frac{1}{n}\left(\sum_{i=1}^{n} P_i^T P_i u^k - \sum_{i=1}^{n} P_i^T y_{j_i^{k+1}}\right), \tag{8}$$

where

$$j_i^{k+1} = \arg\min_j \frac{1}{2}\|P_i u^k - y_j\|^2 - \varphi_j^{k+1}. \tag{9}$$

In (8), notice that $\sum_{i=1}^n P_i^T$ corresponds to an uniform patch aggregation. To simplify, we consider periodic conditions for patch extraction, so that $\sum_{i=1}^n P_i^T P_i = p\mathrm{I}$, where $p = s \times s$ denotes the number of pixels in the patches. Hence, from (8) and considering a step size $\eta\frac{n}{p}$, $\eta > 0$, the update of $u$ through gradient descent can be formulated as: $u^{k+1} = (1-\eta)u^k + \eta v^k$, where $v^k = \frac{1}{p}\sum_{i=1}^n P_i^T y_{j_i^{k+1}}$, can be interpreted as the image formed with the patches from the exemplar image $v$ which are the nearest neighbor to the patches of $u^k$ in the sense of (9). The gradient step then mixes the current image $u^k$ with $v^k$. In the case $\varphi = 0$, the minimum in (9) is reached by associating to each patch of $u^k$ its nearest neighbor in the set $\{y_1, \ldots, y_n\}$ patches of $v$, which exactly corresponds to the texture synthesis algorithm proposed in [20].

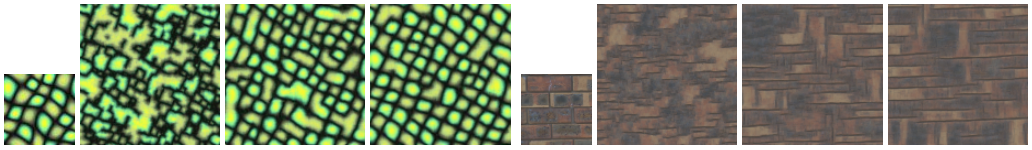


Figure 1: Influence of patch-size $s$ for the single-scale method (Algorithm 1 with $L = 1$).

This image synthesis process is illustrated in Figure 1, with a comparison of image synthesis for different patch sizes. This method cannot take into account variations that may occur at scales larger than $s$, a limitation which is overcome by the multi-scale extension introduced in the next section.

### 3.4 Multi-scale Texture Generation

In the patch-based literature, a common way to deal with multi-scale is to create a pyramid of down-sampled and blurred images. Let $S_l$ be an operator that creates a down-sampled and blurred version of an image for a scale $l \in \{1, ..., L\}$. The image at scale $l$ is of size $n/2^{l-1} \times n/2^{l-1}$ and denoted $u_l = S_l(u)$. The multi-scale texture synthesis is then obtained by minimizing

$$\mathcal{L}(u) = \sum_{l=1}^{L} \max_{\varphi} f_{v^l}(u^l, \varphi), \quad \text{where} \quad f_{v^l}(u^l, \varphi) = \frac{1}{n}\sum_{j=1}^{n} \min_i \left[c(P_j u^l, P_i v^l) - \varphi_i\right] + \frac{1}{m}\sum_{i=1}^{m} \varphi_i. \tag{10}$$

This loss is the sum of OT costs defined on different scales of $u$. As for the single-scale case, an alternate scheme is considered to minimize the loss $\mathcal{L}$ in (10). Considering smooth operators $S_l$, with differential at $u$ given by $D_u S_l(u)$, the gradient descent update of $u$ relies on the following relation combining different scales:

$$\nabla_u \mathcal{L}(u) = \sum_{l=1}^{L} \left(D_u S_l(u)\right)^T \nabla_u f_{v^l}(u^l, \varphi_l^\star) = \sum_{l=1}^{L} G_l(u, \varphi_l^\star). \tag{11}$$

### 3.5 Experiments

The multi-scale synthesis process is presented in Algorithm 1. Figure 2 shows some examples of synthesized textures and comparisons with a patch-based method [20] and a deep learning one [8] based on VGG-19 features [28]. The evolution of the loss function is also drawn to illustrate the numerical stability of the optimization scheme. The value $f(\varphi^{k+1}, u^k)$, which is an approximation of $\sum_l W_2^2(\mu_{u_l^k}, \nu_l)$ between the distributions of patches from $u^k$ and $v$ at all scales $l = 1 \cdots L$, is

almost always decreasing along the iterations $k$. While it is already known [23] that the approach of [8] might have color inconsistencies, it mostly suffers here from the small resolution of the input for which it is difficult to extract deep features. Additionally, one can observe that contrarily to our method and [8], the approach of [20] does not rely on statistics and does not respect the distribution of features from the original sample. Therefore, it must be initialized with a good guess (permutation of patch) instead of any random image.

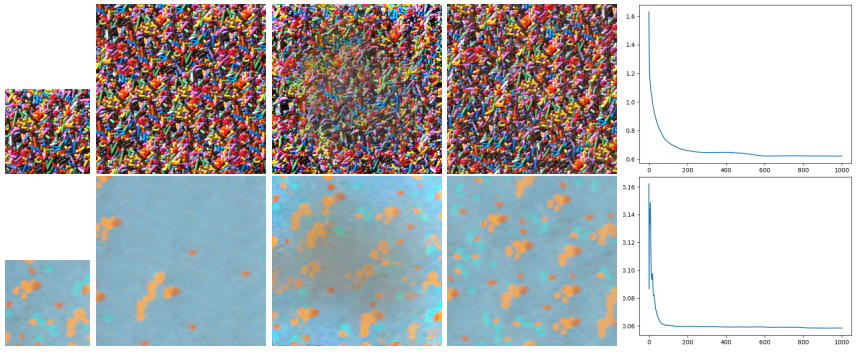


Figure 2: Texture synthesis from a $128 \times 128$ sample by image optimization. Comparison of our multi-scale approach using $s = 4$ (see Alg. 1) with Kwatra [20] (patch size ranging from $s = 32$ to $s = 8$) and Gatys [8] (VGG-19 features). See text for details.

The framework proposed for texture synthesis can be extended to texture inpainting, by taking the patches outside a masked area as the target ones. By optimizing only the pixels within the masked area, the very same algorithm yields an efficient inpainting method, as illustrated in Figure 3.

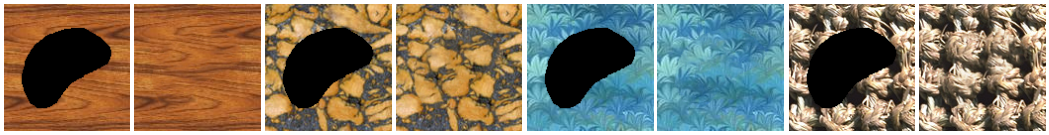

Figure 3: Texture inpainting results: Alg. 1 is ran on the missing part, target patches are from outside.

**Discussion** As previously mentioned, the proposed approach can be seen as a generalization of the iterative texture synthesis method [20]. The visual results are competitive with the state-of-the-art of patch-based synthesis methods. The process requires 1 second to run one iteration for a $256 \times 256$ image on a GPU Nvidia K40m, and while 150 iterations are enough to reach a good visual result, as illustrated by convergence plots in Figure 2. However, the whole optimization has to be done each time a new image is synthetized. In order to define a versatile algorithm that can generate new samples on the fly, we now rely on continuous generative models and convolutional neural networks.

**Algorithm 1** Multi-scale Texture synthesis

**Input:** target image $v$, initial image $u_0$, learning rates $\eta_u$ and $\eta_\varphi$, number of iterations $N_u$ and $N_\varphi$, number of scales $L$
**Output:** image $u$
$u \leftarrow u_0$ and $\varphi_l = 0$ for $l = 1 \dots L$
**for** $k = 1$ **to** $N_u$ **do**
  **for** $l = 1$ **to** $L$ **do**
    **for** $j = 1$ **to** $N_\varphi$ **do**
      $\varphi_l \leftarrow \varphi_l + \eta_\varphi g_l(\varphi_l, S_l(u))$
    **end for**
    $G_l(u, \varphi_l) \leftarrow (D_u S_l(u))^T \nabla_u f_{v_l}(\varphi_l, S_l(u))$
  **end for**
  $u \leftarrow u - \eta_u(k) \sum_{l=1}^{L} G_l(u, \varphi_l)$
**end for**

**Algorithm 2** Multi-scale Convolutional Texture generation with stochastic gradient descent

**Input:** target image $v$, initial weight $\theta_0$, learning rate $\eta_\theta$, number of iterations $N_u$ and $N_\varphi$, number of scales $L$
**Output:** generator parameters $\theta$
**for** $k = 1$ **to** $N_u$ **do**
  **for** $l = 1$ **to** $L$ **do**
    compute $\varphi_l^\star$ with ASGA
    sample $z$ from $\zeta$
    $G_l(\theta) \leftarrow \nabla_\theta f_{v_l}(\varphi_l^\star, g_\theta(z)_l)$
  **end for**
  $\theta \leftarrow \theta - \eta_\theta(k) \sum_{l=1}^{L} G_l(\theta)$
**end for**

# 4 Generative Network

In this section, we consider the problem of training a network to generate images that have a prescribed patch distribution at multiple scales.

## 4.1 The semi-discrete formulation

Let us consider a generator $g_\theta$, parameterized by $\theta \in \Theta \subset \mathbf{R}^d$ that goes from a latent space $\mathcal{Z}$, assumed to be compact, to the space of images. Given a random vector $Z$ of distribution $\zeta$ in the latent space $\mathcal{Z}$, the generated random image is $g_\theta(Z)$, and has distribution $g_\theta \sharp \zeta$ (defined for any Borel set $A$ by $g_\theta \sharp \zeta(A) = \zeta(g_\theta^{-1}(A))$). In the following, we will assume that for $d\zeta$-almost all $z$, $\theta \mapsto g_\theta(z)$ is differentiable and we denote by $J_\theta g_\theta(z)$ the corresponding Jacobian matrix. The patches of the output image are distributed according to $\mu_\theta = \frac{1}{n}\sum_{i=1}^n P_i \sharp g_\theta \sharp \zeta$. One can assume that $\mu_\theta$ is absolutely continuous w.r.t. the Lebesgue measure (this is indeed the generic case). Since the target distribution $\nu$ is still discrete, computing $W_p(\mu_\theta, \nu)$ falls into the semi-discrete case of OT. The semi-dual formulation of OT then writes

$$\mathrm{OT}_c(\mu_\theta, \nu) = \max_{\varphi} F(\varphi, \theta) \quad \text{where} \quad F(\varphi, \theta) = \mathbf{E}_{Z\sim\zeta}\left[\frac{1}{n}\sum_{i=1}^n \varphi^c(P_i g_\theta(Z)) + \frac{1}{m}\sum_{j=1}^m \varphi_j\right] \tag{12}$$

Note that $F$ can be related to the function $f$ we introduced in the previous section, since $F(\varphi, \theta) = \mathbf{E}_{Z\sim\zeta}\left[f(\varphi, g_\theta(Z))\right]$. Therefore we can consider the gradients of $f$ w.r.t. $u$ or $\varphi$ as stochastic gradients for $F$. Indeed, from Theorem 2 we get that $g(\varphi, g_\theta(Z))$ is a stochastic super-gradient w.r.t. $\varphi$, and from Theorem 3 we get $\nabla_\theta f(\varphi, g_\theta(z)) = J_\theta\left(g_\theta(z)\right)^T \nabla_u f(\varphi, u = g_\theta(z))$, which exists as soon as $\varphi, g_\theta(z)$ satisfies the hypothesis of Theorem 3. Notice that, except in degenerate cases, this hypothesis is likely to be true (*e.g.* for the Euclidean cost, it will be true as soon as $g_\theta \sharp \zeta$ is absolutely continuous w.r.t. the Lebesgue measure).

In this semi-discrete framework of OT, $\theta \mapsto \mathrm{OT}_c(\mu_\theta, \nu)$ is expected to be smoother than in the discrete case. We do not provide a precise result here. However, if $\theta$ is a point of differentiability, the envelope theorem ensures that $\nabla_\theta \mathrm{OT}_c(\mu_\theta, \nu) = \nabla_\theta \mathbf{E}[f(\varphi^\star, g_\theta(Z))]$ where $\varphi^\star$ is an optimal dual variable for the OT from $\mu_\theta$ to $\nu$, and if we can differentiate under the expectation, we get $\nabla_\theta \mathrm{OT}_c(\mu_\theta, \nu) = \mathbf{E}[\nabla_\theta f(\varphi^\star, g_\theta(Z))]$.

## 4.2 Proposed algorithm

All the previous considerations together with the multiscale formalism we defined in Section 3 lead us to propose the Algorithm 2 for minimizing the following loss w.r.t. the parameters $\theta$

$$L(\theta) = \sum_{l=1}^{L} \max_{\varphi_l} \mathbf{E}_{z\sim\zeta}\left[f_{v^l}(\varphi_l, g_\theta(z)^l)\right]. \tag{13}$$

In practice, for each iteration $k$ and at each layer we first update $\varphi_l$ with an averaged stochastic gradient ascent (ASGA) as proposed in [9]. Then we sample an image and perform a stochastic gradient step in $\theta$. The fully convolutional neural network designed for texture generation in [29] is taken for $g_\theta$. The main advantage of convolutional networks is that, once learnt, they can generate arbitrarily large images. In our PyTorch implementation, we use the Adam optimizer to estimate the parameters $\theta$. We run the algorithm for $10000$ iterations with a learning-rate $0.01$. An averaged stochastic gradient ascent (ASGA) with $100$ inner iterations is used for computing $\varphi^\star$. In this setting, 30' are required to train our generator with a GPU Nvidia K40m.

## 4.3 Experimental results and discussions

Figure 4 proposes a comparison of our results with four relevant synthesis methods from the literature. We first consider the Texture Networks method [29], which consists in training a generative network using VGG-19 features maps computed on a sample texture. Note that the very same CNN architecture has been used for our model. We also compare to SinGAN [27], a recent GAN technique to generate images from a single example relying on patch sampling, PSGAN [2] a previous approach that applies the GAN framework to the learning of a CNN and Texto [22] which also constrains patch distribution

with OT. We used the official Pytorch implementations of SINGAN (`github.com/tamarott/SinGAN`), PSGAN (`github.com/zalandoresearch/famos`) and Texture Networks (`github.com/JorgeGtz/TextureNets_implementation`), with their default parameters. Additional analysis of the results and comparisons with state of the art methods are provided in the supplementary material.

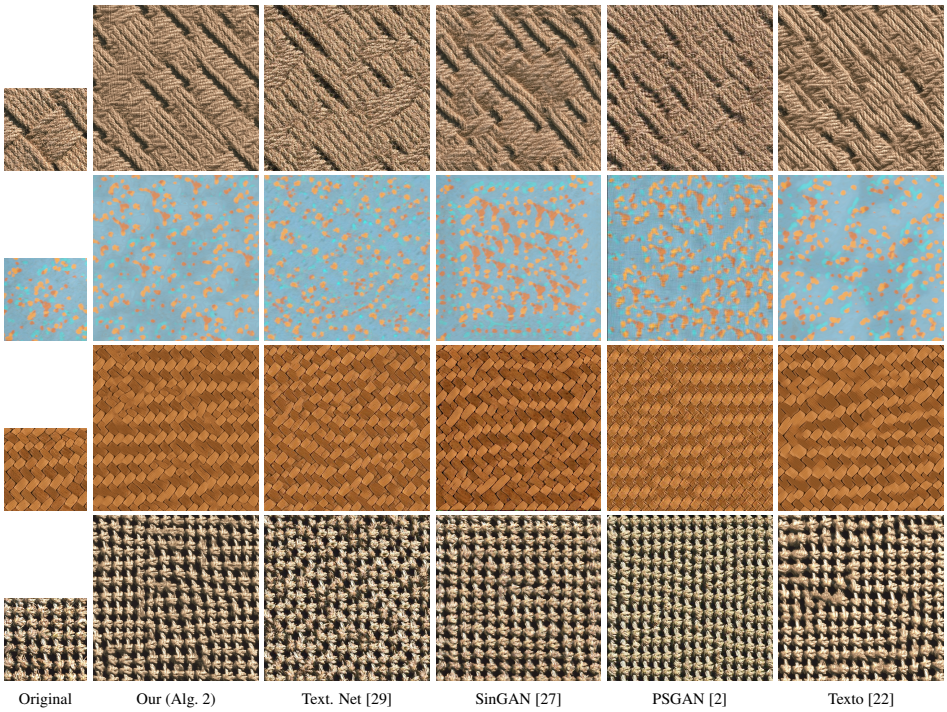


Figure 4: Texture synthesis from a generative network trained on a single $256 \times 256$ sample. Comparison of our multi-scale approach using $4 \times 4$ patches (see Alg. 2) with [29] (using VGG-19 features), SinGAN [27], PSGAN [2] and Texto [22].

The results obtained with our method are visually close to the ones from Texto [22] which also minimizes OT distance between patch distribution. However, the patch-aggregation step from Texto makes the results blurrier than our method since we overcome the aggregation issue by design. Although Texture networks [29] produce textures that look sharper than our results, they may fail to reconstruct larger structures as in the fourth image. Observe as well that patch-based networks are less likely to create visual artifacts (checkerboard patterns, false colors, etc).

## 5 Conclusion

In this paper, we propose an original image generation framework from a single sample, which combines three popular topics in image processing and machine learning: patch-based methods, optimal transport and deep generative models. For both pixel optimization and convolutional network optimization, we achieve results comparable to the state-of-the-art in texture synthesis. Dealing only with the patch distribution of the synthesized textures through OT costs, the pixel optimization method relates to a classical patch-based model [20]. In many patch-based synthesis methods, the different scales and the aggregation of the patches are treated separately. The strength of our approach then comes from the integration of the different synthesis steps into a single gradient descent on the image. Notice that we have restricted conditions regarding the number of target patches on the existence of the functional gradient. However, this is not a limitation for considered applications and numerical experiments show a clear convergence. Still considering the optimal transport framework, we then propose to learn a convolutional neural network for texture synthesis. Our method produces images of arbitrary size, which patch-distributions at various scales are close in 2-Wasserstein distance to the ones of the reference image used for training. This is a significant advantage over most methods in the literature that makes use of the VGG-19 deep network [28] to extract features maps that requires a more complex optimization and a larger memory capacity.

## References


[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223, 2017.

[2] U. Bergmann, N. Jetchev, and R. Vollgraf. Learning texture manifolds with the periodic spatial gan. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 469–477. JMLR. org, 2017.

[3] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65. IEEE, 2005.

[4] Y. Chen, M. Telgarsky, C. Zhang, B. Bailey, D. Hsu, and J. Peng. A gradual, semi-discrete approach to generative network training via explicit wasserstein minimization. *arXiv preprint arXiv:1906.03471*, 2019.

[5] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *IEEE International Conference on Computer Vision*, page 1033, 1999.

[6] B. Galerne and A. Leclaire. Texture inpainting using efficient gaussian conditional simulation. *SIAM Journal on Imaging Sciences*, 10(3):1446–1474, 2017.

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

[8] L. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *NIPS*, pages 262–270, 2015.

[9] A. Genevay, M. Cuturi, G. Peyré, and F. Bach. Stochastic optimization for large-scale optimal transport. In *Advances in neural information processing systems*, pages 3440–3448, 2016.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[11] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

[12] J. Gutierrez, B. Galerne, J. Rabin, and T. Hurtut. Optimal patch assignment for statistically constrained texture synthesis. In *Scale-Space and Variational Methods in Computer Vision*, 2017.

[13] J. Gutierrez, J. Rabin, B. Galerne, and T. Hurtut. On demand solid texture synthesis using deep 3d networks. In *Computer Graphics Forum*. Wiley Online Library, 2018.

[14] P. Henzler, N. J. Mitra, and T. Ritschel. Learning a neural 3d texture space from 2d exemplars. *arXiv preprint arXiv:1912.04158*, 2019.

[15] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[16] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[17] A. Kaspar, B. Neubert, D. Lischinski, M. Pauly, and J. Kopf. Self tuning texture optimization. In *Computer Graphics Forum*, volume 34, pages 349–359. Wiley Online Library, 2015.

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] S. Kolouri, P. E. Pope, C. E. Martin, and G. K. Rohde. Sliced wasserstein auto-encoders. In *International Conference on Learning Representations*, 2019.

[20] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example-based synthesis. In *ACM SIGGRAPH 2005 Papers*, pages 795–802. 2005.

[21] M. Lebrun, A. Buades, and J.-M. Morel. A nonlocal bayesian image denoising algorithm. *SIAM Journal on Imaging Sciences*, 6(3):1665–1688, 2013.

[22] A. Leclaire and J. Rabin. A fast multi-layer approximation to semi-discrete optimal transport. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 341–353. Springer, 2019.

[23] G. Liu, Y. Gousseau, and G. Xia. Texture synthesis through convolutional neural networks and spectrum constraints. In *Int. Conf. on Pattern Recognition (ICPR)*, pages 3234–3239. IEEE, 2016.

[24] L. Mescheder, A. Geiger, and S. Nowozin. Which training methods for gans do actually converge? *arXiv preprint arXiv:1801.04406*, 2018.

[25] J. Portilla and E. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–70, 2000.

[26] F. Santambrogio. Optimal transport for applied mathematicians. *Progress in Nonlinear Differential Equations and their applications*, 87, 2015.

[27] T. R. Shaham, T. Dekel, and T. Michaeli. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4570–4580, 2019.

[28] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[29] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: feed-forward synthesis of textures and stylized images. In *Proc. of the Int. Conf. on Machine Learning*, volume 48, pages 1349–1357, 2016.

[30] G. Xia, S. Ferradans, G. Peyré, and J. Aujol. Synthesizing and Mixing Stationary Gaussian Texture Models. *SIAM Journal on Imaging Sciences*, 7(1):476–508, 2014.

[31] N. Yu, C. Barnes, E. Shechtman, S. Amirghodsi, and M. Lukác. Texture mixer: A network for controllable synthesis and interpolation of texture. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12164–12173, 2019.

[32] Y. Zhou, Z. Zhu, X. Bai, D. Lischinski, D. Cohen-Or, and H. Huang. Non-stationary texture synthesis by adversarial expansion. *arXiv preprint arXiv:1805.04487*, 2018.

[33] S. Zhu, Y. Wu, and D. Mumford. Filters, random fields and maximum entropy (FRAME): Towards a unified theory for texture modeling. *International Journal of Computer Vision*, 27(2):107–126, 1998.

[34] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *2011 International Conference on Computer Vision*, pages 479–486. IEEE, 2011.