A Stochastic Multi-layer Algorithm for Semi-discrete Optimal Transport with Applications to Texture Synthesis and Style Transfer

Arthur Leclaire[1] · Julien Rabin[2]

Abstract

This paper investigates a new stochastic algorithm to approximate semi-discrete optimal transport for large-scale problem, i.e., in high dimension and for a large number of points. The proposed technique relies on a hierarchical decomposition of the target discrete distribution and the transport map itself. A stochastic optimization algorithm is derived to estimate the parameters of the corresponding multi-layer weighted nearest neighbor model. This model allows for fast evaluation during synthesis and training, for which it exhibits faster empirical convergence. Several applications to patch-based image processing are investigated: texture synthesis, texture inpainting, and style transfer. The proposed models compare favorably to the state of the art, either in terms of image quality, computation time, or regarding the number of parameters. Additionally, they do not require any pixel-based optimization or training on a large dataset of natural images.

Keywords Optimal transport · Texture synthesis · Patch matching · Image inpainting · Style transfer

1 Introduction

Since its original formulation by Monge [38], the theory of optimal transportation has been very widely developed [52, 63] and has found many applications in computational sciences [45]. Given two probability distributions μ (source) and ν (target), the optimal transport (OT) problem consists in finding a way to transfer the source mass μ onto the target mass ν while minimizing a transportation cost. The solution then defines a distance between μ and ν that is truly sensitive to the distances in the underlying spaces. Thus, OT provides a natural way to compare probability distributions.

Therefore, OT has been used for numerous applications in imaging science. First, applying OT in the color space permits to evaluate the distance between the color distributions of two images, which was used in [51] for image retrieval.

✉ Arthur Leclaire
Arthur.Leclaire@math.u-bordeaux.fr
http://www.math.u-bordeaux.fr/~aleclaire

Julien Rabin
Julien.Rabin@unicaen.fr
http://rabin.users.greyc.fr

[1] University of Bordeaux, IMB, Bordeaux INP, CNRS, UMR 5251, 33-400 Talence, France

[2] ENSICAEN, CNRS, GREYC, Normandie University, Caen, France

The corresponding OT maps provide candidates to transfer the colors from one image to the other, but must be regularized to produce the visually plausible results [48]. Similarly, from two images, one can transfer the colors to a kind of averaged color distribution, which was coined as *midway image equalization* in [7]. Both problems of color transfer and color equalization have found a variational formulation in [44]. In these color applications, the transport maps act on a three-dimensional space for RGB images and on a one-dimensional space for gray-level images. Even before that, a one-dimensional OT distance was used to compare angular descriptors identified as probability distributions on the circle, in order to address image matching [47].

In a higher-dimensional setting, several authors propose to use OT distances to compare images identified as probability distributions. For example, a Kantorovich–Rubinstein norm was used in the data-fidelity term for image denoising in [30]. Transportation maps acting on images can then be used to perform image warping [1,12,22,35,43], which is relevant for images that reflect some kind of density of material. Notice that similar tools can also be used to process shapes identified as probability distributions which allows to perform shape interpolation [57] and shape registration [10] with applications in brain imaging [11].

Several other imaging applications require to use optimal transport in a feature space. Indeed, gathering several features

defined around a pixel gives information on the regularity or textural aspect of the image in the neighborhood. Such a point of view was already developed in [51] for image retrieval. The authors of [49] proposed to address image segmentation by constraining the distribution of features (color and derivatives norms) in the segmented regions. More generally, it is possible to use OT distances between feature distributions, or even image distributions, for texture modeling. In [64], the OT distance between Gaussian texture models was related to the distance between Fourier transform modulii, which led to an elegant formulation of Gaussian texture mixing. The authors of [58] proposed to use OT distances between distributions of linear and nonlinear features (e.g., wavelet responses or dictionary-based features) to address texture synthesis and restoration. In the present paper, considering that the textural aspect is encoded in the patch distribution, we will exploit OT between patch distributions to address texture synthesis.

A common bottleneck with many imaging applications of the OT framework is that, except in one dimension, solving an OT problem is usually difficult and computationally expensive. Indeed, in the case of two discrete distributions, the OT problem is equivalent to a linear programming problem. If the distributions support have the same number of points n, this problem can be solved with the Hungarian algorithm [28] which has complexity $\mathcal{O}(n^3)$ and thus scales quite badly with n. One possibility to overcome this numerical issue is to add an entropic penalization in the OT problem, leading to the definition of Sinkhorn distances [6,27,50]. Such regularized OT distances can be computed with a fixed point algorithm that is geometrically convergent. However, for some imaging applications, the entropic regularization may deteriorate the OT plan. And besides, the computational gain is only available for the case of two discrete distributions with n points, whereas for high-dimensional OT problems, the discretization of the underlying space is prohibited (representing 100-dimensional distributions with ten bins in each direction would require to store a discrete vector of size 10^{100}).

In this paper, we will focus on the semi-discrete case of OT, meaning that the source distribution μ is assumed to have a density with respect to the Lebesgue measure, while the target distribution ν is discrete. The authors of [2] have shown that in this case, the OT map takes the form of a weighted nearest neighbor (NN) assignment where the weight parameters solve a differentiable concave optimization problem. Since [2], several numerical solutions have been proposed to solve the semi-discrete OT problem [25,26,31,37], but they often require the exact computation of a gradient which amounts to computing the μ-measure of polytopes. In a high-dimensional setting, computing such integrals may be intractable. Thus, one may turn to stochastic gradient algorithms to approximate the OT maps, as proposed in [19] and later studied in [3]. But when the number J of points in the tar-

get distribution is large, this stochastic algorithm converges slowly [15].

A classical way to improve the practical convergence of optimization methods is to exploit a multiscale representation of the data. For which concerns optimal transport, numerous multiscale strategies have already been proposed which essentially consists in working on simplified versions of the measures μ and ν. Several authors [5,21,42] have proposed to discretize the measures on a grid and then use a grid refinement strategy to target the true OT map. They propose different arguments to demonstrate the stability of the refined transport plans (which exploits directly or indirectly the sparsity of the true OT plan when μ and ν are absolutely continuous). In the case of large-scale discrete OT, the author of [53,54] proposed multiscale algorithms for unregularized and regularized OT problems that exploit a hierarchical representation of the supports of the source and target distributions in order to drastically diminish the number of variables in the dual formulation of OT. This coarse-to-fine strategy was also used in [11] to accelerate computations on very large datasets. However, for semi-discrete OT in a high-dimensional setting, these multiscale schemes are hardly tractable.

Another strategy is to consider a multiscale decomposition $(\nu^\ell)_{0 \le \ell < L}$ of the measure ν, meaning that ν^ℓ is an approximation of ν with a fixed budget of J^ℓ points, $J^0 > J^1 > \cdots > J^{L-1}$ (and similarly for the source μ). This approach was proposed in [37] to solve richer and richer semi-discrete OT problem by initializing the optimization algorithm at one scale with an extrapolation of the solution at the previous scale. A similar approach (with decomposition of both the source and target measures) was followed by [20] to solve large-scale discrete OT problems.

In this paper, we propose a different multiscale approach to approximately solve the semi-discrete OT problem. The main idea is to consider so-called multi-layer transport maps that can be roughly understood as a composition of weighted NN assignments that solve restricted OT problems between μ and the simplified measures ν^ℓ (obtained with a hierarchical clustering of ν at different resolutions). In general, such a multi-layer transport map may not exactly solve the OT problem because a simple weighted NN assignment may not be decomposed as a multi-layer map. In other words, restricting to such multi-layer maps induces a bias that can be related to the geometry of the hierarchical clustering of ν. However, optimizing such a multi-layer transport map amounts to solving many semi-discrete OT problems with much smaller target distributions. For that reason, when using stochastic algorithms for semi-discrete OT, we demonstrate that nearly optimal cost can be attained in a faster way with multi-layer transport maps (even if the total number of parameters of a multi-layer map is slightly larger than a weighted NN assignment).

Such a multi-layer approach thus helps to efficiently solve high-dimensional semi-discrete OT problems linked to imaging applications. In particular, we will be able to improve a texture model [15] based on semi-discrete OT in the patch space. The initial model presented in [15] was limited to target distributions of 1000 patches of size 3×3. Using multi-layer transport maps will allow to consider richer distributions of patches with larger size, thus enriching the class of well-reproduced texture. We will see that these new texture models can also be used for texture inpainting and style transfer.

The rest of the paper is organized as follows. In Sect. 2, we recall the framework of semi-discrete optimal transport. Section 3 is devoted to multi-layer transport maps. First, we define multi-layer transport maps and analyze their optimality conditions. And then, we propose a stochastic algorithm to optimize multi-layer transport maps and examine its behavior on a simple one-dimensional example. In Sect. 4, we demonstrate that such multi-layer transport maps can be used to improve a texture model based on semi-discrete OT. We also show that this texture model can be used to address textural inpainting. Finally, in Sect. 5, we propose to extend the framework used in texture synthesis to address style transfer. Let us mention that a preliminary version of this work has been published as a conference paper [29]. Compared to the conference version, we contributed with a more thorough study of optimality conditions for multi-layer transport maps, a more extensive study of the performance of these models in texture synthesis and new applications on textural inpainting and style transfer. The experiments shown in this paper can be reproduced with the MATLAB implementation available at https://www.math.u-bordeaux.fr/~aleclaire/texto/multilayer.php.

2 Semi-discrete Optimal Transport

Let μ, ν be two probability measures on $\mathbb{R}^D$. For the sake of simplicity, we will restrict to the case of the quadratic cost, even if some of the concepts developed in this paper could be used with other cost functions. For the quadratic cost, Monge's formulation of OT consists in solving

$$\inf_T \int_{\mathbb{R}^D} \|x - T(x)\|^2 \mathrm{d}\mu(x), \qquad \text{(OT-M)}$$

where the infimum is taken over all measurable functions $T : \mathbb{R}^D \to \mathbb{R}^D$ for which $T_\sharp \mu = \nu$, where $T_\sharp \mu$ is the pushforward measure defined by $T_\sharp \mu(A) = \mu(T^{-1}(A))$ for every Borel set $A \subset \mathbb{R}^d$. The Monge problem admits a convex relaxation due to Kantorovich

$$W^2(\mu, \nu) = \inf_{\Pi} \int_{\mathbb{R}^D \times \mathbb{R}^D} \|x - y\|^2 \mathrm{d}\Pi(x, y), \qquad \text{(OT-K)}$$

where the infimum is taken on all probability distributions Π on $\mathbb{R}^D \times \mathbb{R}^D$ with marginal μ, ν. If (OT-M) admits a solution T, then $(\mathrm{Id} \otimes T)_\sharp \mu$ is a solution to (OT-K). But the Kantorovich problem is more general in the sense that they may exist no map T such that $T_\sharp \mu = \nu$. Besides, under some conditions (that will be satisfied in the semi-discrete case), one can show that (OT-M) admits a solution which is uniquely defined almost everywhere. In such a case, the solution will be denoted by T^*. We refer the reader to the books [52,63] for an exhaustive presentation of the OT framework.

In this paper, we only consider the semi-discrete case of optimal transportation. Indeed, we assume that μ is absolutely continuous with respect to the Lebesgue measure on $\mathbb{R}^D$ with density ρ and that ν is supported on a finite set Y

$$\nu = \sum_{y \in Y} \nu(y) \delta_y. \qquad (1)$$

As proved in [2,26,31], the solution of (OT-M) has the form of a weighted NN assignment. Indeed, for $v \in \mathbb{R}^Y$, one can define $T_{Y,v}$ by

$$T_{Y,v}(x) := \underset{y \in Y}{\operatorname{argmin}} \|x - y\|^2 - v(y). \qquad (2)$$

This map is defined almost everywhere (i.e., at all points where the previous argmin is uniquely defined). The preimages

$$\mathcal{L}_v(y) = T_{Y,v}^{-1}(\{y\}) \qquad (3)$$

are called the *Laguerre cells* and form a partition of $\mathbb{R}^D$ up to a negligible set called the *Laguerre tessellation* (or also power diagram).

Then, it is known [2,26] that $T_{Y,v}$ is a solution of (OT-M) as soon as v maximizes the concave function

$$H(v) := \mathbb{E}_{X \sim \mu} [h(X, v)] \qquad (4)$$

where

$$h(x, v) = \left(\min_{y \in Y} \|x - y\|^2 - v(y) \right) + \sum_{y \in Y} v(y)\nu(y). \qquad (5)$$

One can see that for all $v \in \mathbb{R}^J$ and for almost all $x \in \mathbb{R}^D$,

$$\partial_{v(y)} h(x, v) = -\mathbb{1}_{\mathcal{L}_v(y)}(x) + \nu(y) \qquad (6)$$

where $\mathbb{1}_{\mathcal{L}_v(y)}$ is the indicator function of $\mathcal{L}_v(y)$, and that

$$\partial_{v(y)} H(v) = \mathbb{E}_{X \sim \mu} \left[\partial_{v(y)} h(X, v) \right]$$
$$= -\mu(\mathcal{L}_v(y)) + \nu(y). \qquad (7)$$

Several authors [2,26,31,37] have proposed exact gradient-based methods or quasi-Newton schemes in order to optimize the weights v when the distributions μ and ν are defined on $\mathbb{R}^D$ for $D = 2$ or $D = 3$ dimensions, in cases where the μ-measure of the Laguerre cells needed in (7) is tractable. However, in a high-dimensional framework where such integrals are not tractable, one may turn to the average stochastic gradient descent (ASGD) Algorithm 1 to solve it. One can show [3,15,19] that this algorithm has a convergence guarantee in $\mathcal{O}(\frac{\log k}{\sqrt{k}})$ (in expectation). Numerical experiments also confirm its slow convergence rate, especially when the number J of points in the target distribution gets very large.

Algorithm 1: ASGD to estimate OT map $T_{Y,v}$, solution of (OT-M)

1: **Inputs:** source density μ, target discrete distribution ν, initial assignment weight (*e.g.,* $\tilde{v} = 0$ and $v = 0$), and gradient step (*e.g.,* $C = 1$)
2: **for** $k = 1, 2, \ldots$ **do**
3:   Draw a sample $x \sim \mu$
4:   Compute the gradient $g \leftarrow \nabla_v h(x, \tilde{v})$ (see Eq. (6))
5:   Gradient ascent of weights: $\tilde{v} \leftarrow \tilde{v} + \frac{C}{\sqrt{k}} g$
6:   Average of updates: $v \leftarrow \frac{k-1}{k} v + \frac{1}{k} \tilde{v}$
7: **return** v

---

3 Multi-layer Semi-discrete Transport Maps

In order to cope with the slow convergence of the stochastic algorithm used to estimate the semi-discrete OT map, we propose to approximate the optimal map with hierarchical applications of several weighted NN assignments that are tuned to solve simpler semi-discrete OT problems (i.e., with smaller target distributions). These smaller problems are related to a multiscale decomposition $\nu^0, \ldots, \nu^{L-1}$ of the target measure, that is, a collection of measures that "summarizes" ν from fine ($\ell = 0$) to coarse ($\ell = L$) resolution.

As already mentioned in the introduction, a multiscale algorithm for semi-discrete OT has already been proposed in [37]. However, as reported in [29] and illustrated in experimental Sect. 3.6, it is not straightforward to design an "upscaling" scheme that gives a good initial guess for the variable v at one scale from the solution found at the previous scale. In the following, we thus propose a new strategy which consists in:

1. modeling the OT map itself as a multiscale hierarchical operator;
2. optimizing at all scales simultaneously.

In order to avoid confusion with aforementioned multiscale techniques, we refer to the proposed hierarchical model as a *multi-layer* transport map.

3.1 Decomposition of the Target Measure

In all this section, we will work with a decomposition

$$\nu^0, \ldots, \nu^{L-1}, \nu^L \tag{8}$$

of the target measure. By convention, $\nu^0 = \nu$ and ν^L will be supported on a singleton, so that there are only L non-trivial scales. At scale $\ell \in \{0, \ldots, L\}$, we will use a measure

$$\nu^\ell = \sum_{y \in Y^\ell} \nu^\ell(y) \delta_y \tag{9}$$

supported by a finite set Y^ℓ of cardinal J^ℓ (which is a prescribed budget of points at scale ℓ, with $J^0 = J$, and $J^L = 1$).

Following [37], this decomposition is built recursively: the measure $\nu^{\ell+1}$ should be a close approximation of ν^ℓ with a budget of $J^{\ell+1}$ points. More precisely, given ν^ℓ, $\nu^{\ell+1}$ is built as an approximate solution of

$$\min_m W^2(\nu^\ell, m) \tag{10}$$

where the min is taken on all discrete measures m on $\mathbb{R}^D$ whose support has $J^{\ell+1}$ points.

This non-convex problem is known to be equivalent to a weighted K-means problem, which we approximately solve with Lloyd's algorithm. At the end of this algorithm, we get a set $Y^{\ell+1} \subset \mathbb{R}^D$ of $J^{\ell+1}$ centroids corresponding to a partition

$$Y^\ell = \bigsqcup_{y \in Y^{\ell+1}} C_y^\ell \tag{11}$$

and the associated masses

$$\forall y \in Y^{\ell+1}, \quad \nu^{\ell+1}(y) = \nu^\ell(C_y^\ell). \tag{12}$$

Additionally, we consider normalized distributions per cluster

$$\forall y \in Y^{\ell+1}, \forall z \in C_y^\ell \quad \tilde{\nu}_y^\ell(z) = \frac{\nu_y^\ell(z)}{\nu_y^\ell(C_y^\ell)}. \tag{13}$$

With the convention $J^L = 1$, the clustering of Y^{L-1} is trivial with only one cluster, and the corresponding centroid (the only point of Y^L) is the ν^{L-1}-barycenter of Y^L.

An illustration of such a multiscale decomposition is given in Fig. 1. Let us emphasize that the previous decomposition
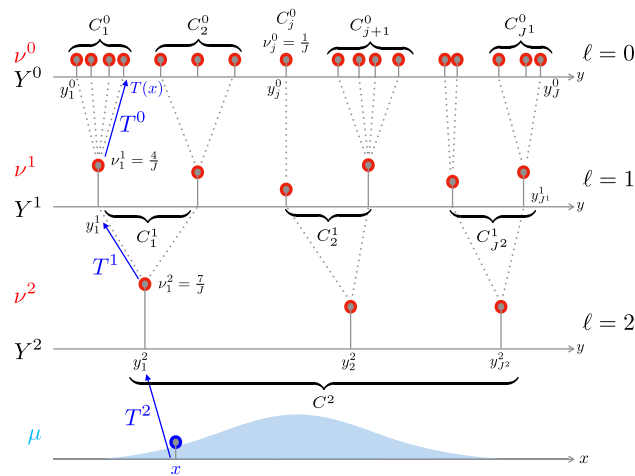
Fig. 1 Illustration of the multiscale approximation of the discrete target distribution ν in the 1D case and the multi-layer transport map $T(x)$ applied sequentially with $L = 3$ layers. See the text for more details about notation

algorithm (based on K-means) can be called hierarchical in the sense that there exists a tree structure on the elements of the sets Y^ℓ, (Each $y \in Y^\ell$ has exactly one parent in $Y^{\ell+1}$.) This hierarchical structure will be crucial for the construction of multi-layer transport maps.

3.2 Multi-layer Transport Maps

Now, we can define multi-layer transport maps by using a hierarchical tree search based on weighted NN assignments.

Definition 1 Let us denote by $\mathbf{Y}$ the hierarchical clustering of Y composed of the sets $(Y^0, Y^1, \ldots, Y^L)$ and the clusters $(C_y^\ell)_{0 \leq \ell < L, y \in Y^{\ell+1}}$ satisfying (11). Let us also consider the multi-layer parameters

$$\mathbf{v} = \left(v_y^\ell \right)_{0 \leq \ell < L, y \in Y^\ell} \in \prod_{\ell=0}^{L-1} \mathbb{R}^{Y^\ell}. \tag{14}$$

Then, we can define a multi-layer map $T_{\mathbf{Y},\mathbf{v}}$ recursively as follows. Let us fix $x \in \mathbb{R}^d$. We set $T^L(x) = y^L$ (only point in Y^L). And then, for $\ell = L-1, \ldots, 0$, denoting $y = T^{\ell+1}(x)$, we set

$$T^\ell(x) = T_{C_y^\ell, v_y^\ell}(x) = \operatorname*{argmin}_{z \in C_y^\ell} \|x - z\|^2 - v_y^\ell(z). \tag{15}$$

Then, $T_{\mathbf{Y},\mathbf{v}}(x) = T^0(x)$ (Fig. 2).

3.3 Optimal Multi-layer Maps

Let $T_{\mathbf{Y},\mathbf{v}}$ be a multi-layer map, and we recall the definition T^ℓ ($\ell = L, \ldots, 0$) of the intermediate maps. Notice that applying these intermediate maps at a point x amounts to tracing

back a hierarchy of Laguerre cells to which x belongs. Therefore, the sets $\mathcal{L}_y^\ell = (T^\ell)^{-1}(\{y\})$ provide a decomposition

$$\mathbb{R}^D = \bigsqcup_{y \in Y^\ell} \mathcal{L}_y^\ell \tag{16}$$

which is a partition up to a negligible set. These subsets are obtained by intersecting Laguerre cells in a nested way; therefore the $\mathcal{L}_y^\ell$ will be called the nested cells.

In the following, we denote by $\mu_{|A}$ the restriction of the measure μ to the Borel set A.

Definition 2 A multi-layer map $T_{\mathbf{Y},\mathbf{v}}$ associated with the hierarchical clustering $\mathbf{Y}$ is said to be optimal if, for all $0 \leq \ell < L$ and all $y \in Y^{\ell+1}$, $T_{C_y^\ell, v_y^\ell}$ realizes the semi-discrete OT from $\mu_{|\mathcal{L}_y^{\ell+1}}$ to $\nu_{|C_y^\ell}^\ell$.

Remark Let us emphasize that this optimality condition must be understood in a coarse to fine manner. Indeed, for a given scale $0 \leq \ell < L$, the conditions $\mu(\mathcal{L}_y^{\ell+1}) = \nu^\ell(C_y^\ell)$, i.e., $\mu(\mathcal{L}_y^{\ell+1}) = \nu^{\ell+1}(y)$ are ensured for all $y \in Y^{\ell+1}$ if and only the map $T^{\ell+1}$ realizes the semi-discrete OT at the previous scale. By uniqueness of optimal semi-discrete OT maps, it follows that there exists a unique optimal multi-layer map (up to a μ-negligible set).

Alternately, for a given nested cell $\mathcal{L}_y^{\ell+1}$, $y \in Y^{\ell+1}$ at the previous scale, one can consider the normalized measure

$$\tilde{\mu}_y^{\ell+1} = \frac{\mu_{|\mathcal{L}_y^{\ell+1}}}{\mu(\mathcal{L}_y^{\ell+1})}. \tag{17}$$

Then, from the last remark, we get that a multi-layer map $T_{\mathbf{Y},\mathbf{v}}$ is optimal if and only if for all $0 \leq \ell < L$ and all $y \in Y^{\ell+1}$, $T_{C_y^\ell, v_y^\ell}$ realizes the semi-discrete OT from $\tilde{\mu}_y^{\ell+1}$ to $\tilde{\nu}_y^\ell$.

It is also possible to express the optimality of multi-layer maps in terms of the variables $\mathbf{v}$: For all $0 \leq \ell < L$ and all $y \in Y^{\ell+1}$, the weights v_y^ℓ should maximize the function

$$\begin{aligned} H_y^\ell(v_y^\ell) = &\int_{\mathcal{L}_y^{\ell+1}} \min_{z \in C_y^\ell} (\|x - z\|^2 - v_y^\ell(z)) \mathrm{d}\mu(x) \\ &+ \sum_{z \in C_y^\ell} v_y^\ell(z) \nu^\ell(z) \end{aligned} \tag{18}$$

Notice that the weights $\mathbf{v}$ should solve, at each layer ℓ, $J^{\ell+1}$ subproblems of semi-discrete OT where the nested cells $\mathcal{L}_y^{\ell+1}$ intervene.

Proposition 1 *The multi-layer map $T_{\mathbf{Y},\mathbf{v}}$ solves the semi-discrete OT problem between μ and ν if and only if*
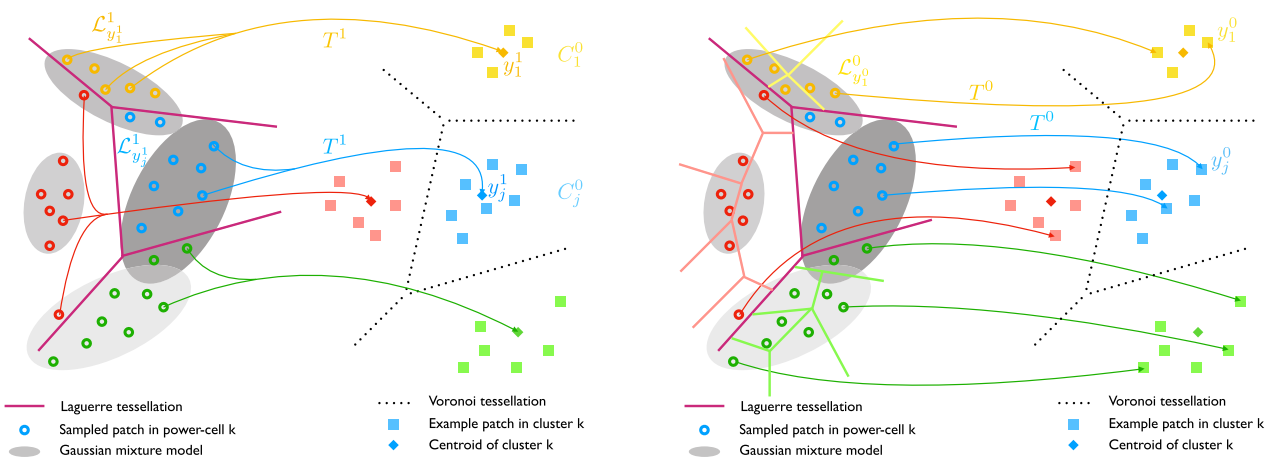
First layer $\ell = 1$: the coarse transport map T^1 maps every points of a Laguerre cell j of μ to the centroid of the corresponding cluster C_j^0 in ν.

Second layer $\ell = 0$: the fine mapping $T = T^0$ is defined from the transport maps $T_{C_j^0, v_j^\ell}$.

Fig. 2 Illustration of a multi-layer map (for $L = 2$ layers). Here the source distribution μ is chosen to be a Gaussian mixture model with 4 components (in gray levels). For each layer ℓ, the arrows illustrate

$T^\ell(x)$ (arrows) the multi-layer mapping of samples x drawn from μ (circle points on the left) to the points of the discrete distribution ν^ℓ (diamonds for layer $\ell = 1$ and square for layer $\ell = 0$)

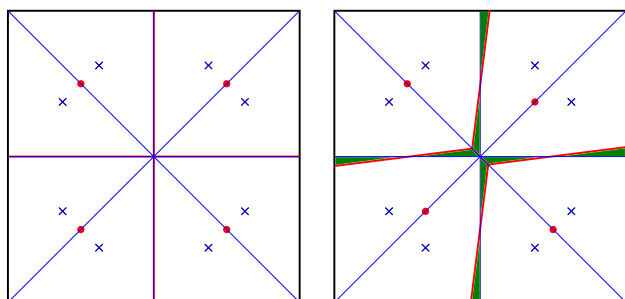$$\mathcal{L}_y^{\ell+1} = \bigcup_{z \in C_y^\ell} \mathcal{L}_z^\ell. \tag{20}$$

$\square$

Fig. 3 Bias due to the hierarchical clustering. In this figure, we illustrate the bias which is induced by the hierarchical clustering and quantized by (27). Here, μ is the uniform distribution on the largest square, and ν the discrete uniform distribution on the eight blue crosses. The red points form a possible clustering at level 1. We draw the optimal Laguerre cells with blue boundaries and the optimal Laguerre cells at level 1 with red boundary. For the very particular case of the left diagram, the OT map is exactly the optimal multi-layer map. However, on the right diagram, the hierarchical induces a bias at level 1 that can be measured by the area of the region colored in green (Color figure online)

One practical consequence of Proposition 1 is that, if $T_{\mathbf{Y},\mathbf{v}}$ is optimal, then the boundary of a nested cell $\mathcal{L}_y^\ell$ is included in the reunion of boundaries of $(T^*)^{-1}(\{z\})$ for all $z \in Y$ that are children of y. But, except for very particular cases (see Fig. 3), this has no reason to happen because each face composing the boundary of $\mathcal{L}_y^\ell$ is orthogonal to one of the segments joining two points of Y^ℓ. Therefore, the geometry of the partition $(\mathcal{L}_y^\ell)_{y \in Y^\ell}$ at scale ℓ is very much impacted by the positions of the centroids $y \in Y^\ell$. In other words, the choice of hierarchical clustering imposes a bias that cannot be coped with by the optimization of the weights $\mathbf{v}$.

In dimension $D = 1$, the situation is much simpler, as shown by the next proposition.

Proposition 2 *Assume that the dimension $D = 1$. Let us also assume that the hierarchical clustering is increasing in the following sense: For all $0 \le \ell < L$, for all $y_1, y_2 \in Y^{\ell+1}$ such that $y_1 < y_2$ then we have $z_1 < z_2$ for all $z_1 \in C_{y_1}^\ell$ and $z_2 \in C_{y_2}^\ell$. Then, the optimal multi-layer map realizes the semi-discrete OT from μ to ν.*

$$\forall y \in Y, \quad \mathcal{L}_y^0 = (T^*)^{-1}(\{y\}) \tag{19}$$

up to a μ-negligible set. In this case, at each scale ℓ, all the nested cells $\mathcal{L}_y^\ell$ can be written as a reunion of Laguerre cells associated with the OT map T^*, up to a μ-negligible set.

Proof First of all, let us recall that with the adopted assumptions on μ, the OT map T^* is uniquely defined μ-almost everywhere. It follows that $T_{\mathbf{Y},\mathbf{v}} = T^*$ a.e. is equivalent to (19). The last statement holds because, by construction of the maps T^ℓ, for all $0 \le \ell < L$ and all $y \in Y^{\ell+1}$,

Proof Let us show by induction on $\ell = L - 1, \ldots, 0$ that T^ℓ is non-decreasing and realizes the semi-discrete OT between μ and ν^ℓ. By Definition 2 and since there is only one point in Y^L (for which $\mathcal{L}_y^L = \mathbb{R}^D$ and $C_y^{L-1} = Y$), T^{L-1} is the OT map between μ and ν^{L-1}. Next, assume

that $T^{\ell+1}$ is non-decreasing and realizes the OT between μ and $\nu^{\ell+1}$. Again, from Definition 2, and for all $y \in Y^{\ell+1}$, on $\mathcal{L}_y^{\ell+1}$, T^ℓ coincides with $T_{C_y^\ell, \nu_y^\ell}$ which realizes the OT from $\mu_{|\mathcal{L}_y^{\ell+1}}$ to $\nu_{|C_y^\ell}^\ell$, and which is thus non-decreasing. Thus, T^ℓ is a non-decreasing map such that for each $z \in Y^\ell$, $\mu((T^\ell)^{-1}(\{z\})) = \nu^\ell(z)$. This implies that T^ℓ is the semi-discrete OT map between μ and ν^ℓ. $\qquad\square$

Remark The above discussion shows that, except in dimension 1, fixing a hierarchical clustering of ν imposes constraints on the shapes of the optimal nested cells. Optimizing both the multi-layer maps and the hierarchical clustering of ν seem worth of interest, but it would probably lead to a much more complex non-convex problem.

3.4 Stochastic Optimization

As already said, finding an optimal multi-layer map amounts to compute in a coarse-to-fine manner many semi-discrete OT maps, by solving the restricted semi-dual problems (18). If we have optimality at the previous scales $\ell' > \ell$, then the OT problem at scale ℓ is well-defined. Then, going back to the Monge formulation of these separates subproblems, the map T^ℓ actually minimizes

$$\int_{\mathbb{R}^D} \|T^\ell(x) - x\|^2 \mathrm{d}\mu(x) = \sum_{y \in Y^{\ell+1}} \int_{\mathcal{L}_y^{\ell+1}} \|T^\ell(x) - x\|^2 \mathrm{d}\mu(x) \tag{21}$$

but with marginal constraints on each of the Laguerre cells $\mathcal{L}_y^{\ell+1}$.

If we consider the concave problem (18), the gradient of H_y^ℓ can still be computed

$$\frac{\partial H_y^\ell}{\partial v_y^\ell(z)} = -\mu(\mathcal{L}_y^{\ell+1} \cap \mathcal{L}_z^\ell) + \nu^\ell(z) \tag{22}$$

where we kept the notation $\mathcal{L}_y^{\ell+1} \cap \mathcal{L}_z^\ell$ for $\mathcal{L}_z^\ell$ to emphasize that $\mathcal{L}_z^\ell$ is a subset of $\mathcal{L}_y^{\ell+1}$ which is fixed by the previous layer. If v_y^ℓ is a critical point of H_y^ℓ, then

$$\forall z \in C_y^\ell, \quad \mu(\mathcal{L}_y^{\ell+1} \cap \mathcal{L}_z^\ell) = \nu^\ell(z) \tag{23}$$

which implies that $\mu(\mathcal{L}_y^{\ell+1}) = \nu^\ell(C_y^\ell)$. However, this last condition is not guaranteed if we do not have optimality at the previous scales. In this case, H_y^ℓ have no critical point and thus no maximum.

For that reason, we can only propose a heuristic algorithm to optimize the multi-layer map $T_{\mathbf{Y},\mathbf{v}}$. It consists in performing gradient ascent to simultaneously increase the values of all functions H_y^ℓ for all layers $0 \le \ell < L$ and all $y \in Y^{\ell+1}$. However, in order to cope with the fact that the OT maps

are not optimal at the previous layers, we consider instead the cost adapted to the normalized measures defined in (17) and (13)

$$\tilde{H}_y^\ell(v_y^\ell) := \mathbb{E}_{X \sim \tilde{\mu}_y^{\ell+1}} \left[\tilde{h}_y^\ell(X, v_y^\ell) \right] \tag{24}$$

with

$$\tilde{h}_y^\ell(x, v_y^\ell) = \left(\min_{z \in C_y^\ell} \|x - z\|^2 - v_y^\ell(z) \right) + \sum_{z \in C_y^\ell} v_y^\ell(z) \tilde{v}_y^\ell(z). \tag{25}$$

The corresponding gradient

$$\frac{\partial \tilde{h}_y^\ell}{\partial v_y^\ell(z)}(x, v_y^\ell) = -\mathbb{1}_{\mathcal{L}_z^\ell}(x) + \tilde{v}_y^\ell(z) \tag{26}$$

is normalized as well (i.e., it has zero sum after taking the expectation) and is used in lieu of the former gradient estimate Eq. (22). The corresponding optimization procedure is summarized in Algorithm 2.

Algorithm 2: ASGD for the estimation of the multi-layer map $T_{\mathbf{Y},\mathbf{v}}$.

Inputs: source density μ, target distribution ν, gradient step C, number of layers L and number of iterations T

1: Hierarchical clustering $\{\nu^0, \dots, \nu^{L-1}\}$ of ν
2: Set $\tilde{v}_y^\ell \,\forall \ell, y$ using Eq. (13) (normalized measures)
3: Set $\tilde{v}_y^\ell \leftarrow 0$, $\forall \ell, y$ (weights initialization)
4: Set $n_y^\ell \leftarrow 0$, $\forall \ell, y$ (number of visits in cluster C_y^ℓ)
5: **for** $t = 1, \dots, T$ **do**
6: $\quad$ Draw a sample $x \sim \mu$
7: $\quad$ **for** $\ell = L - 1, \dots, 0$ **do**
8: $\quad\quad$ Using $T^{\ell+1}(x)$, compute the cluster index $y = T^\ell(x)$ (Eq. (15))
9: $\quad\quad$ $n_y^\ell \leftarrow n_y^\ell + 1$
10: $\quad\quad$ $g \leftarrow \nabla_{\tilde{v}_y^\ell} \tilde{h}_y^\ell(x, \tilde{v}_y^\ell)$ (Eq. (26))
11: $\quad\quad$ $\tilde{v}_y^\ell \leftarrow \tilde{v}_y^\ell + \frac{C}{\sqrt{n_y^\ell}} g$
12: $\quad\quad$ $v_y^\ell \leftarrow v_y^\ell + \frac{1}{n_y^\ell}\left(\tilde{v}_y^\ell - v_y^\ell\right)$

Outputs: $\{\nu^\ell\}_{0 \le \ell < L}$ and $\mathbf{v} = \left(v_y^\ell\right)_{0 \le \ell < L, y \in Y^\ell}$

3.5 Aggregating the Errors

In order to measure how much a multi-layer map drifts from the true OT map T^*, one may essentially distinguish two types of errors. The first one is the bias that is induced by a fixed hierarchical clustering and which cannot be coped with the optimization of $\mathbf{v}$, as discussed in Sect. 3.3. Denoting by $\mathcal{L}_y^{*\ell}$ the nested cells of the optimal multi-layer map, it is possible to quantize this bias at level ℓ by

$$\text{bias}(\ell) = \sum_{y \in Y^\ell} \mu \left(\mathcal{L}_y^{*\ell} \bigcup_{z \in C(y)} (T^*)^{-1}(\{z\}) \right) \qquad (27)$$

where AB refers to the symmetric difference between the sets A, B, and where $C(y)$ is the set of children of y at scale 0. In Fig. 3, we illustrate this bias on a simple two-dimensional example. In general, computing this bias is a difficult problem since it requires to know the optimal nested cells $\mathcal{L}_y^{*\ell}$, which is equivalent to know the optimal multi-layer map. However, in dimension 1, this bias is known to be zero by Proposition 2.

Another source of error comes from the suboptimality of the elementary transport maps $T_{C_y^\ell, v_y^\ell}$ parameterized by the vectors v_y^ℓ. Of course, one could directly consider the values of the functions H_y^ℓ. But it is certainly more revealing to consider the L^1-norm of their gradients (22) and to aggregate them. Therefore, we obtain an error

$$E = \sum_{\ell=0}^{L-1} \sum_{z \in Y^\ell} |\mu(\mathcal{L}_z^\ell) - \nu^\ell(z)|. \qquad (28)$$

Analogously to the single-layer case, this corresponds to the amount of mistransported mass at all scales. Given a hierarchical clustering $\mathbf{Y}$, then $\min E = 0$ which, by definition, is attained only for the optimal multi-layer map. Notice also that, contrary to (27), the error (28) can be estimated by a Monte Carlo method.

3.6 1D Experiments

Experimental setting In this section, we propose a simple one-dimensional experiment to evaluate the convergence speed of the Algorithm 2 to approximate the optimal multi-layer map. For all the experiments shown in this section,

- μ is he standard Gauss distribution $\mathcal{N}(0, 1)$,
- ν is the uniform discrete distribution on J equally spaced points between -1 and 1.

The benefit of such a one-dimensional setting is that the OT map between μ and ν can be computed (using the quantiles of μ), and besides, it is easy to compute distances between a transported measure $T_\sharp\mu$ and ν. In the following, we will focus on the Kolmogorov distance $d_{\text{KOL}}(T_\sharp\mu, \nu)$, which is defined as the L^∞ distance between cumulative distribution functions. Another benefit of the one-dimensional setting is that, as shown by Proposition 2, the bias (27) induced by the monotone hierarchical clustering is zero. Therefore, the optimal multi-layer map targeted by Algorithm 2 is exactly the optimal transport map T^*. The budget J^ℓ of points at scale ℓ is chosen manually (see more explanation below), and the hierarchical clustering is computed with Lloyd's algorithm.

Let us mention that in the multi-layer setting, the comparison cannot be fairly performed at a fixed number of iterations since one iteration of ASGD for the one-layer transport has not the same cost as one iteration of ASGD for multi-layer transport. Therefore, the comparison will be based on the true computational time (in seconds).

Multi-layer versus one layer First, let us compare the multi-layer framework with $L = 2$ (Algorithm 2) with the single-layer framework (Algorithm 1) with $J = 10^3$ and 10^4 points in the target distribution. The number of clusters for the 2-layer transport is $J^1 = \lfloor\sqrt{J}\rfloor$ (integer part of $\sqrt{J}$). We also compare to a naive multiscale variant of Algorithm 2 that consists in exploiting another decomposition ν^ℓ of the target measures (with more scales) and estimating the semi-discrete OT from μ to ν^ℓ in a coarse-to-fine manner by initializing the weights v with an extrapolation of the weights found at the previous scale (for example, simply propagating the values from parent to children in the hierarchical clustering). Such an extrapolating rule was also presented in [37] for a deterministic framework (based on an efficient second-order optimization scheme which is not implemented here).

The results can be seen in Fig. 4. One can see that the bilayer algorithm reaches a good value for the distance $d_{\text{KOL}}(T_\sharp\mu, \nu)$ in a faster way than the single-layer algorithm, especially when J gets very large. However, in the case of $J = 1000$ points, it is interesting to notice that, up to a certain time, the single-layer algorithm attains a better cost (even if the bias induced by the hierarchical clustering is exactly zero). This reflects the fact that Algorithm 2 is only a heuristic optimization scheme, and it is expected to have a more oscillatory behavior than the single-layer algorithm. In particular, we observed that the convergence at the fine layers is slower because the current cells at this layer depend on those obtained at the coarsest scales (and setting the cells at the coarsest scale is an instance of a stable convex stochastic optimization algorithm). Thus, it is especially interesting to turn to the multi-layer algorithm if J is very large with a constrained computational time. Notice also that, in the stochastic setting, the naive multiscale procedure does not help much in terms of convergence speed. Besides, in this naive procedure, one should choose a budget of iterations per scale, which is less trivial in the stochastic setting than in the deterministic quasi-Newton setting of [37] (where only
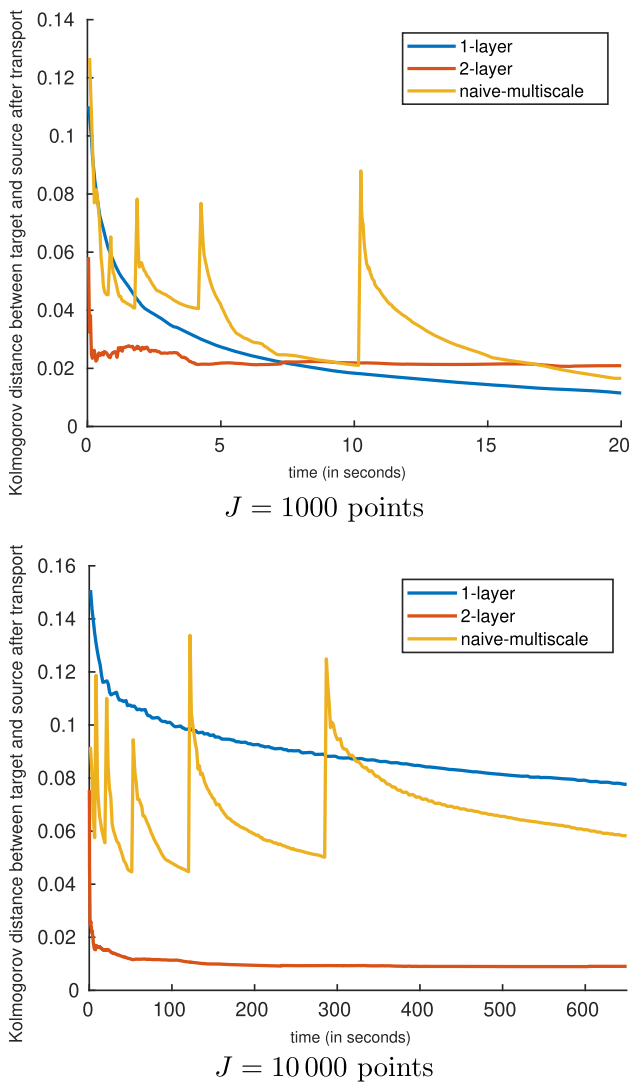
$J = 1000$ points

$J = 10\,000$ points

Fig. 4 Two-layer versus one-layer. We compare with the multi-layer Algorithm 2) (two-layer) with the single-layer algorithm (one-layer) and with the naive multiscale procedure (naive-multiscale). We monitor the Kolmogorov distance between $T_\sharp \mu$ and ν. The horizontal axis represents the computational time (in seconds) and the vertical axis the Kolmogorov distance between the current transported measure $T_\sharp \mu$ and the target measure ν. When J is very large, the two-layer approach leads to a better transport map in a reasonable time

≈ 10 iterations are needed to obtain the solution with very good precision).

Setting the number of clusters We first assume here for simplicity that Y can be decomposed into L scales with balanced clusters. We denote by n_L the number of NN comparison required to evaluate the L-layer transport map $T_{\mathbf{Y},\mathbf{v}}$ (15) at a given point; p_L is the number of weight parameters $v_y^\ell(z)$ describing $T_{\mathbf{Y},\mathbf{v}}$.

For $L = 1$, we have $n_1 = J$ and $p_1 = J$.

For $L = 2$ layers with J_1 balanced clusters and a total of J points, $n_2 = J_1 + \frac{J}{J_1}$: the minimum number of comparison

is then $n_2 = 2\sqrt{J}$ for $J_1 = \sqrt{J}$. The corresponding number of parameters is $p_2 = J_1 + J_1 \times \frac{J}{J_1} = \sqrt{J} + J$.

For $L = 3$ layers with J_1 and J_2 balanced clusters at each layer respectively, the number of points comparison per iteration for J points is $n_3 = J_1 + J_2 + \frac{J}{J_1 J_2}$: the minimum number of comparison is then $n_3 = 3\sqrt[3]{J} < n_2$ for $J_2 = J_1 = \sqrt[3]{J}$. The number of parameters is then $p_3 = J_1 + J_1 \times J_2 + J_1 J_2 \times \frac{J}{J_1 J_2} = \sqrt[3]{J} + \sqrt[3]{J^2} + J > p_2$.

Thus, from the sole perspective of computation load indicated by n_L, one should use a hierarchical representation where the number of clusters per scale is close to $\sqrt[L]{J}$. One could then hope that such a choice would provide a faster convergence, since the number of iteration per second is maximized. This is confirmed empirically in Fig. 5, where we consider the same experimental setting as before, with $J = 10^4$ (Fig. 5a), $J = 10^5$ (Fig. 5b) and $J = 10^6$ (Fig. 5c). In these experiments, the optimal number of clusters for convergence corresponds approximately to $\sqrt[L]{J}$.

Setting the number of layers Observe now that the number of comparisons n_L decreases much faster than the number of parameters p_L increases when using perfectly balanced clusters. Moreover, it is interesting to notice that, while the number of parameters grows with the number of layers, the maximum number of parameters is actually bounded by $2J$. Indeed, even if the following geometric sum is divergent

$$p_L = \sum_{\ell=1}^{L} J^{\frac{\ell}{L}} = \frac{1 - J^{\frac{1+L}{L}}}{1 - J^{\frac{1}{L}}} - 1 \xrightarrow[L \to \infty]{} +\infty$$

the worst practical case, i.e., the deepest possible search tree that can be built, corresponds to the binary classification tree, for which we have (setting $J = 2^L$)

$$p_L = \sum_{\ell=1}^{L} 2^\ell = 2^{1+L} - 2 = 2(J - 1) < 2J$$

but only $n_L = 2L$ comparisons.

Setting aside questions about complexity (such as optimization of memory access, data structure and parallel computing), one would be tempted to conclude that, as the number of layers does not impact much the number of parameters of the model while providing an interesting speed-up, one should use the highest possible number of layers. However, increasing the number of layers makes it more difficult for the estimated transport map to get close to the optimal one. Hence, there is a trade-off between using more layers to reduce the computation time and less layers to reduce the complexity of the model.

Figures 5 and 6 illustrate the impact of increasing the number of layers on convergence speed for $L \in \{1, 2, 3\}$ layers and from $J = 10^4$ to $J = 10^7$. The comparison of performance shows that, even if a larger number of layers allows
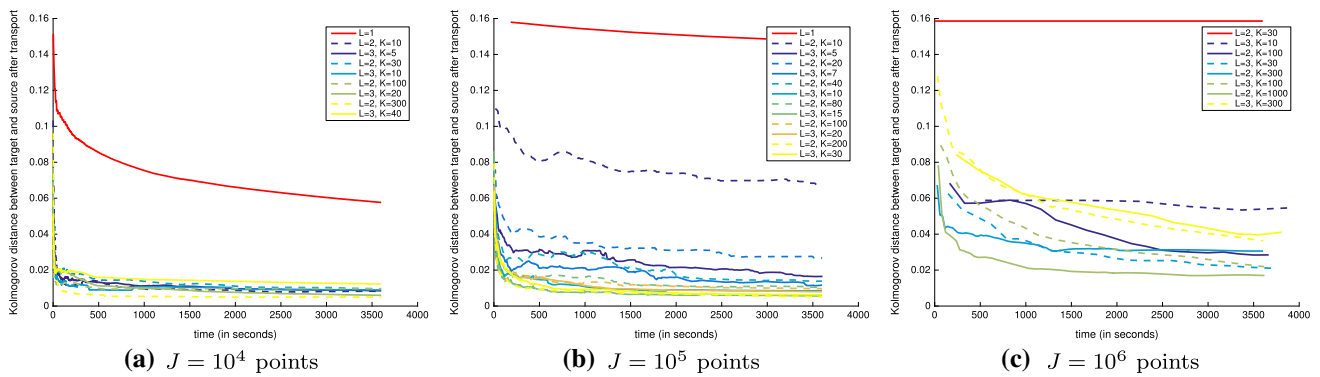
(a) $J = 10^4$ points   **(b)** $J = 10^5$ points   **(c)** $J = 10^6$ points

Fig. 5 Comparison of convergence speed for estimating the multi-layer transportation plan, according to the number L of layers and the number $J^\ell = K$ of clusters. Increasing the number of layers accelerates convergence when the number of clusters is optimal ($K \approx \sqrt[L]{J}$). (Note: each curve is displayed every 10^4 iterations; the initial offset corresponds to the amount of time required to reach such a number of iterations, increasing with K.)

(a) $J = 10^6$ points with $K = 100$ **(b)** $J = 10^7$ points with $K = 30$
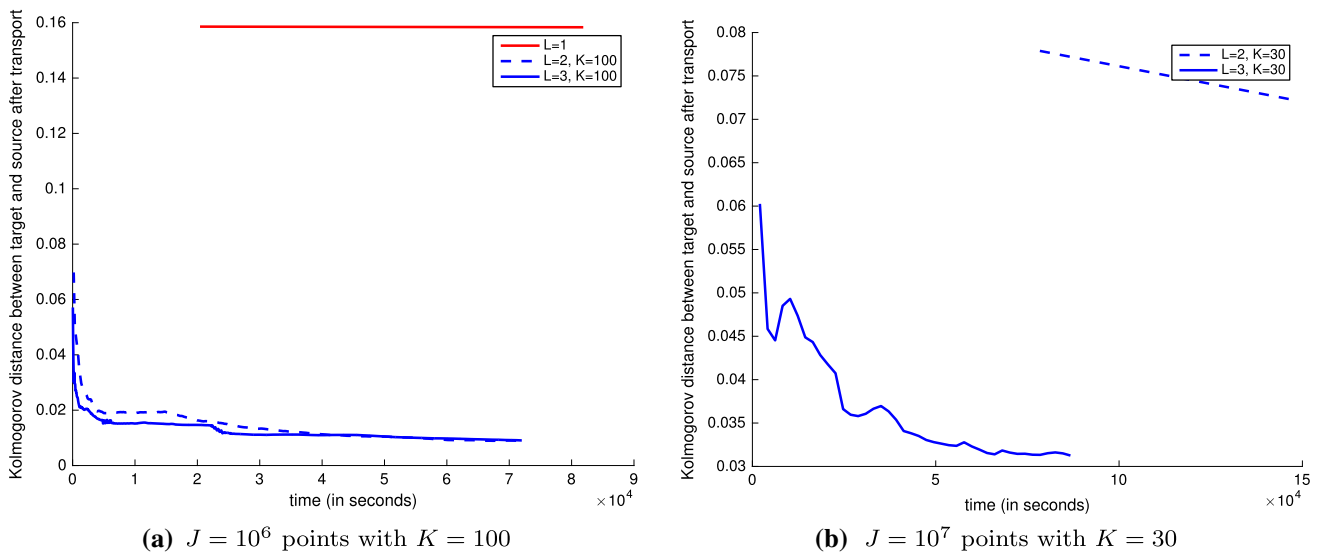
Fig. 6 Comparison of convergence speed for estimating the multi-layer transportation plan for a large number J of points. We see here that increasing the number of layers can improve the convergence speed when J is very large, but it may increase the bias due to hierarchical clustering.

for more samples to be drawn during optimization, the bias error caused by clustering discussed previously in Sect. 3.5 is more difficult to cope with. Nevertheless, the benefit of increasing the number of layers is yet overwhelming when considering a large number of points J (Fig. 6).

4 Application to Texture Synthesis

In this section, we show how the optimal multi-layer maps introduced above can be used to enrich a texture model based on OT in the patch space [15]. One main interest of semi-discrete OT maps for patch-based texture synthesis is that the transport maps project onto patches seen in the exemplar texture while maintaining a global statistical consistency. The main limitation of the model of [15] was that the discrete target patch distributions were constrained to have $\approx 10^3$ points (otherwise, the ASGD algorithm would converge too slowly), which prevents one from using patches larger than 3×3 (e.g., the 7×7 patch distribution of a natural texture needs much more than 10^3 patches to be accurately represented). As was shown in the previous sections, the multi-layer strategy allows to approximate the transport map with larger discrete target distributions and thus to extend our texture model to 7×7 patches, which greatly enlarges the class of well-reproduced textures. Let us emphasize that one must not confound the layers of the multi-layer OT maps of the previous section and the image resolutions; the model here defined will indeed be multiscale in both these aspects.

In all the following, we denote by $u : \Omega \to \mathbb{R}^d$ the original texture (with d channels, $d = 1$ for gray-level images, and $d = 3$ for color images). We also denote by $\omega = \{0, \dots, w-1\}^2$ the patch domain and by $\mathbb{R}^D$ the patch space (where $D = dw^2$).

4.1 Single-Resolution Model

The texture model [15] consists in transforming a stationary Gaussian random field by applying patch transport maps at several resolutions. First, we recall the construction of the texture model for a single resolution.

The single resolution model is built on the Gaussian random field U defined by

$$\forall a \in \mathbb{Z}^2, \ U(a) = \bar{u} + \sum_{b \in \mathbb{Z}^2} t_u(b) W(a - b), \qquad (29)$$

where $\bar{u} = \frac{1}{|\Omega|} \sum_{a \in \Omega} u(a)$, W is a normalized centered Gaussian random field (i.e., the $W(a)$ are independent with standard $\mathcal{N}(0, 1)$ distribution), and where $t_u = \frac{1}{\sqrt{|\Omega|}}(u - \bar{u})\mathbf{1}_\Omega$. Then, we extract all patches $U_{|a+\omega}$ of U, apply to all these patches the same map $T : \mathbb{R}^D \to \mathbb{R}^D$ and then aggregate the transformed patches with a simple average. In other words, we define the transformed random field by

$$\forall a \in \mathbb{Z}^2, \quad V(a) = \frac{1}{|\omega|} \sum_{b \in \omega} T(U_{|a-b+\omega})(b). \qquad (30)$$

In order to reimpose geometric structures of the exemplar texture in a statistically coherent way, the patch map T should solve (at least approximately) the OT between the distribution μ of a patch $U_{|\omega}$ of U (which is a Gaussian distribution with explicit parameters) and the empirical distribution ν of the patches of u. Such a transformed random field is still stationary and possesses some properties that were listed in [15], for example a covariance control and long-range independence.

Therefore, μ is absolutely continuous and ν is discrete so that the OT between μ and ν is actually a weighted NN assignment as in Eq. (2). In our previous work [15], we used such a weighted NN assignment, estimated with Algorithm 1. But again, the estimation step was then very slow for $J \gg 1000$ and so we restricted to $J = 1000$ patches in the target distribution, which constrained us to work only with 3×3 patches. The rest of this section aims at replacing the weighted NN assignment by a multi-layer map and precisely assessing the benefit of such multi-layer maps when working with larger target distributions and larger patches.

Before presenting the results, let us give the details about the remaining parameters of the model. In contrast to our previous work, the target distribution ν is here given by *all* patches of u. Thus, the monolayer transport map T_v is parameterized by a single $v \in \mathbb{R}^J$ where J is the number of patches in u (for example, for a 128×128 image, we have $J \approx 16000$). For the multi-layer transport, we use only $L = 2$ layers (thereby defining bilayer transport maps) and perform a two-scale hierarchical clustering with $J^1 = 40$. (The clusters are found using Lloyd's k-means algorithm.)

In Fig. 7, in the single-resolution case, we compare the synthesized images obtained with a monolayer patch transport and a bilayer patch transport, with patch size $w = 3, 5, 7$. One can first remark that the visual quality of the synthesized texture is very limited with this model working at a single resolution. Indeed, increasing the patch size allows to retrieve larger geometric structures from the exemplar, but $w = 7$ is not large enough to capture all structures of the textures shown in Fig. 7. We will cope with this strong restriction in Sect. 4.2.

However, beyond the geometric content, the patch statistics are better retrieved with a bilayer map than a monolayer map. This reflects again that Algorithm 2 allows to better approximate the transport map in a more reasonable time. Indeed, in this experiment, the number of iterations was set to 10^5 for monolayer transport, and 10^6 for bilayer transport, and yet, the required computational time was much lower in the bilayer case.

The statistical benefit is confirmed by the patch distributions shown in Fig. 8. Since the patch space is high-dimensional ($D = 75$ for color 5×5 patches), we only represent the one-dimensional distributions obtained after projecting on a few principal components of the patch space. In general, on the most important principal components, the multi-layer map will perform at least as good as the monolayer map, (Even if, in some cases, the approximation is not perfect.) Actually, applying the same methodology on a larger set of textures and principal components (not shown here) allows to do draw the following conclusions:

- On a wide majority of components, the multi-layer and monolayer maps perform nearly equally.
- On few main principal components (that will dominate in the visual perception), the multi-layer map better approximates the reference distribution.
- Sometimes (for example, on the last principal components with less energy), the monolayer map performs better.

These remarks correlate the visual differences observed in Fig. 7.

4.2 Multi-resolution Model

In this section, we define a texture model using multi-layer maps at several image resolutions. We will work with S subsampled versions u_s, ($0 \le s \le S - 1$) of the original image at different resolutions, u_s being defined on a subdomain

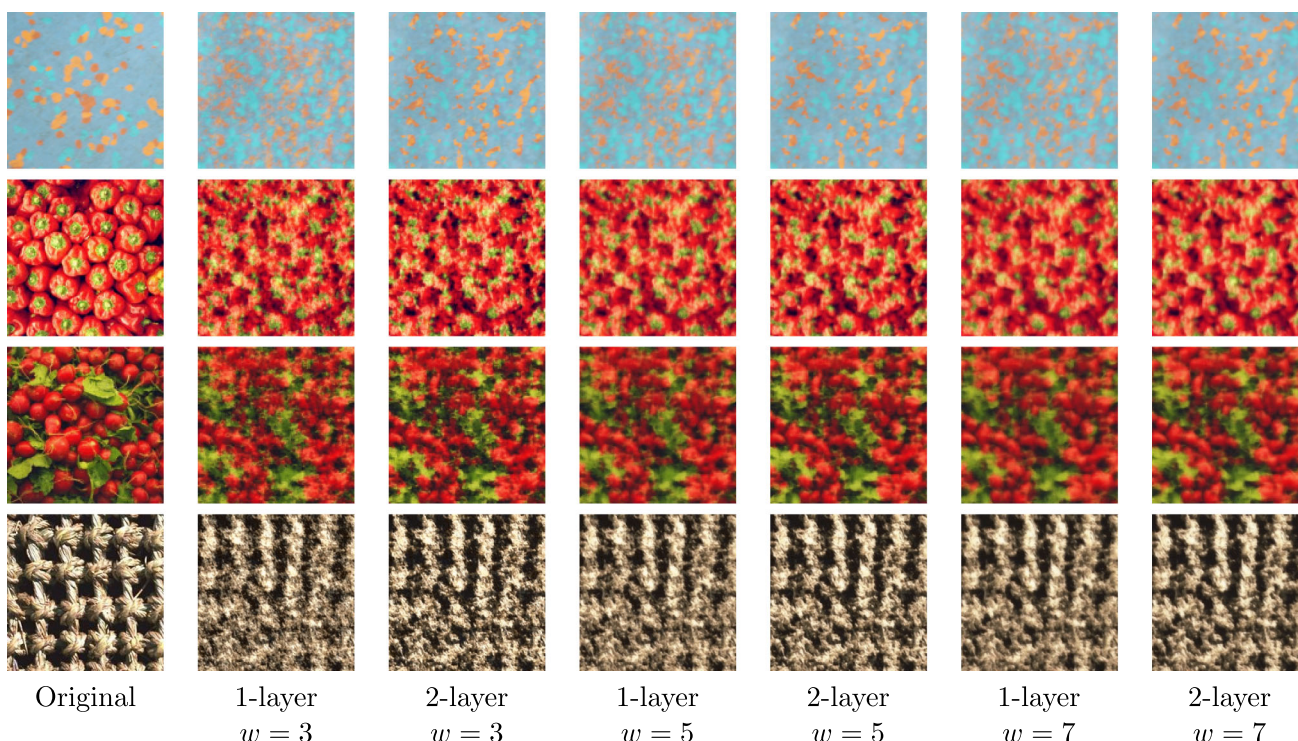| Original | 1-layer $w = 3$ | 2-layer $w = 3$ | 1-layer $w = 5$ | 2-layer $w = 5$ | 1-layer $w = 7$ | 2-layer $w = 7$ |

Fig. 7 Single-resolution model, synthesis results. For several original textures shown in the first column, we compare several synthesis results obtained by the single-resolution transformed random field (30), where T is either a monolayer or bilayer transport map and where the patch size w ranges from 3 to 7. Even if the visual differences between the monolayer and bilayer map are subtle in terms of geometric content, one can see that the patch statistics (and in particular the color distribution) are better respected with the bilayer transport maps

$\Omega_s \subset 2^s \mathbb{Z}^2$. We will also denote by ν_s the empirical distribution of $w \times w$ patches of u_s.

The model is defined by successive patch transport and exemplar-based upsampling as summarized in Algorithm 3. The estimation of the model can be done during one initial pass of synthesis (that can be performed offline). At the coarsest resolution $s = S - 1$, as in the previous section, the model is initialized with the Gaussian random field U_{S-1} associated with u_{S-1}. Then, for every scale $s = S - 1, \dots, 1$, a patch transform T_s is applied to all patches of the current synthesis. This transformation is computed during the estimation stage with the two following steps:

- Fit a GMM distribution μ_s to the patches of U_s.
- Compute the patch transformation T_s that realizes the OT from μ_s to ν_s.

Once estimated, all patches are transformed with T_s and averaged, which defines a transformed random field

$$V_s(a) = \sum_{b \in 2^s \omega} T_s(U_{s|a-b+2^s \omega})(b), \ x \in 2^s \mathbb{Z}^2. \quad (31)$$

Since T_s is actually a patch assignment, there is a "coordinate map" $C_s : 2^s \mathbb{Z}^2 \to \Omega_s$ which allows to express V_s as

$$V_s(a) = \sum_{b \in 2^s \omega} u_s(C_s(a - b) + b), \ x \in 2^s \mathbb{Z}^2. \quad (32)$$

Then, an exemplar-based upsampling step allows to initialize the synthesis at the next resolution. It consists in taking twice larger patches at the same positions. This is equivalent to set for all $a \in 2^s \mathbb{Z}^2$ and for all $t \in \{0, 2^{s-1}\}^2$,

$$U_{s-1}(a + t) = \sum_{b \in 2^s \omega} u_{s-1}(C_s(a - b) + b + t). \quad (33)$$

An illustration of this coarse-to-fine synthesis procedure is available in [15].

Here again, instead of taking a single-layer transport map for T_s, we propose to use a bilayer transport map that better approximates the semi-discrete OT between μ_s and ν_s, even for larger patches. Therefore, in contrast to [15] (referred to as a single-layer model or "1-layer" in the captions), the model will not be limited to 3×3 patches anymore. Indeed, in the experiments shown in this section, the bilayer maps are defined on 7×7 patches.
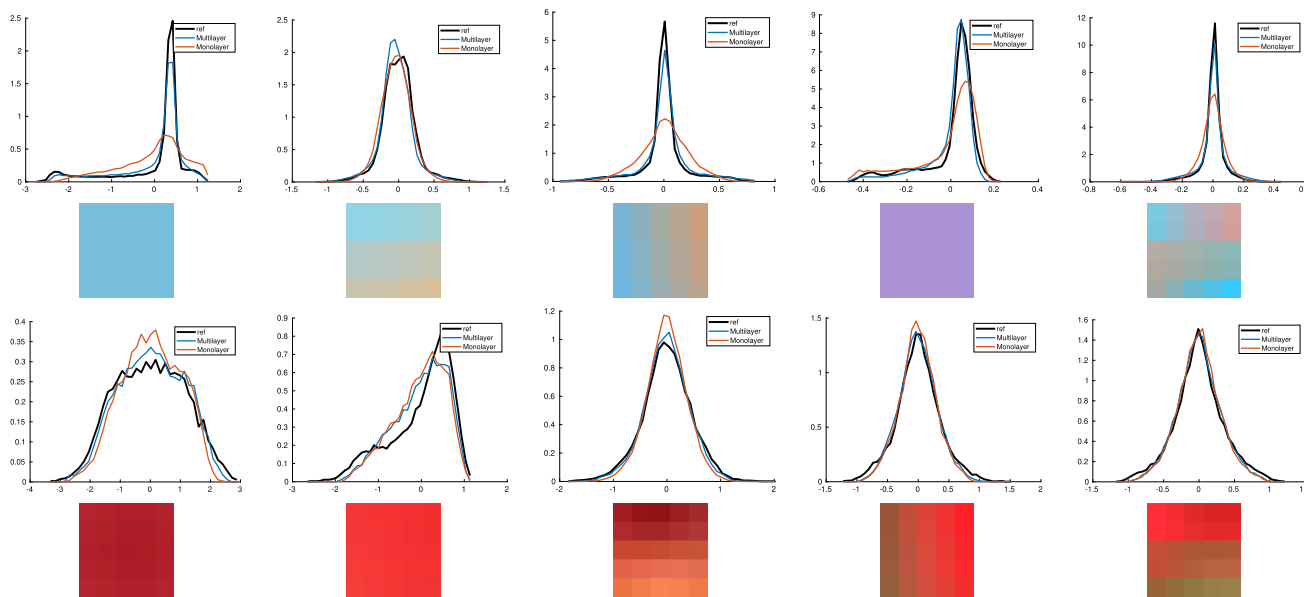
Fig. 8 Single-resolution model, output patch distributions. In this figure, we examine the output patch distributions after the monolayer and bilayer transport maps for the single-resolution model with patch size $w = 5$, for the textures shown in the two first rows of Fig. 7. Each diagram represents the one-dimensional distribution obtained after projecting on a principal axis in the patch space, and the corresponding patch principal component is displayed just below. On each diagram, the black curve is the reference distribution, the blue curve is the distribution after multi-layer transport, and the yellow curve is the distribution after monolayer transport. This experiment reflects again the fact that in a large-scale setting, multi-layer maps provide better transportation maps which here help to better preserve patch distributions compared to monolayer maps

---

Algorithm 3: Texture analysis/synthesis

1: *Inputs:* input texture u
2: *Initialization:* Sample U_{S-1} (29).
3: **for** $s = S - 1, \ldots, 0$ **do**
4:   **if** Analysis **then**
5:     · Fit GMM μ_s to the patches of U_s.
6:     · Estimate transport T_s from μ_s to ν_s
        (with stochastic Algorithm 2)
7:   Apply patch transport T_s on patches of U_s
8:   Aggregate patches to get V_s (31)
9:   **if** $s > 1$, upsample V_s to get U_{s-1} (33)
10: **return** V_0

Let us give the parameters of the used bilayer maps. In all the examples shown below, we used $S = 4$ scales for images smaller than 500×500 and $S = 6$ scales for larger images. Since the target patch distribution ν_s of the exemplar texture has a more complex structure for coarse scales, the number of clusters J^1 is adapted to the scale, i.e., we take $J^1 = 10$ for $s = S-1$, $J^1 = 20$ for $s = S-2$, and $J^1 = 40$ for $s < S-2$. The hierarchical clustering is again found using the k-means algorithm; however, in order to keep a reasonable complexity even if the exemplar texture is very large, we fix a budget B of

patches in each cluster. If the number of patches in a cluster C_y^0 exceeds B, then we randomly select B patches in this cluster. This budget also depends on the scale: $B = 100$ for $s = S-1$, $B = 200$ for $s = S-2$ and $B = 400$ for $s < S-2$. At the finest scales, the global budget is thus $40 \times 400 = 16000$; this selection step amounts to simplifying ν with only 16000 patches. For which regards the source distribution, during the estimation phase, a GMM distribution μ_s with 10 components is fitted to the current patch distribution of U_s using the Expectation Maximization algorithm [36]. Using 10 Gaussian components to model the 7×7 patch distribution of a texture seems relevant in comparison with [15] where 4 components were used to model the 3×3 patch distribution of a texture and also to [65] where ≈ 20 components were used to model the 8×8 patch distribution of a natural image. Finally, in order to reimpose details at the finest resolution, we apply on V_0 a last bilayer transport map on 3×3 patches to get the output image V.

Let us emphasize on the fact that the overall algorithm can be decoupled into an estimation and synthesis step. In other words, the estimation of bilayer transport maps at all resolutions (which is quite costly) can be done once and for all. Once estimated, these transport maps can be applied to all patches and also be used to synthesize many images with arbitrary size. For this reason, if the model is pre-estimated, then the synthesis algorithm has a very low computational
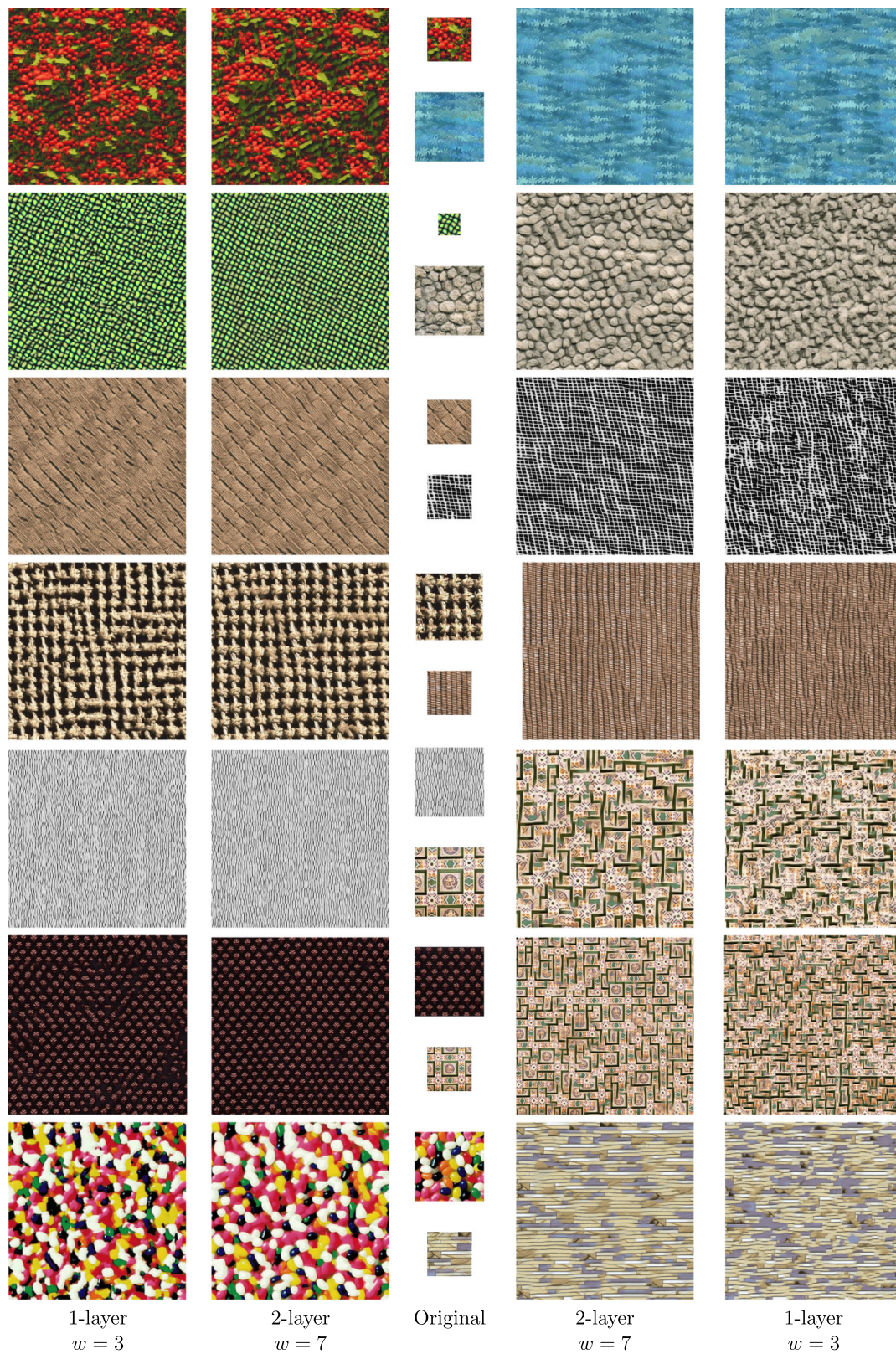
Fig. 9 Multi-resolution model, synthesis results. For several exemplar textures displayed in the middle column, we present several synthesis results obtained with a multi-resolution model exploiting multi-layer transport maps working on 7×7 patches at each resolution. We compare with the previous model [15] based on single-layer transport on 3×3 patches. See the text for comments
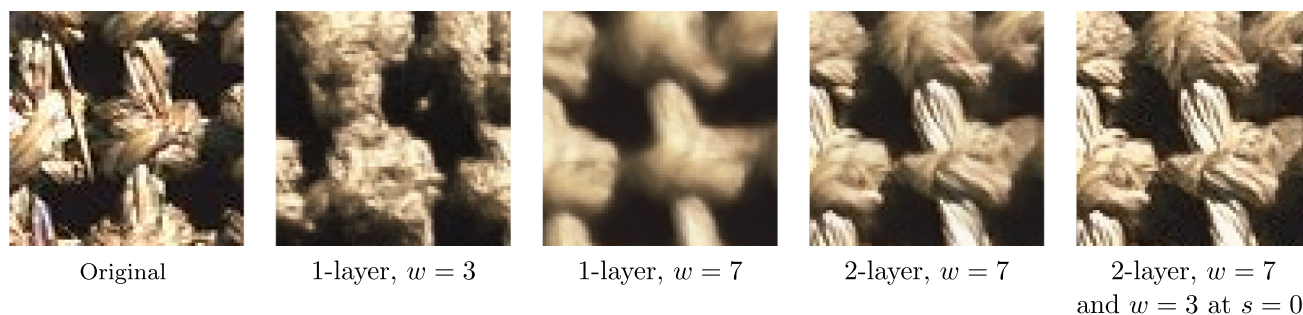
| 1-layer $w = 3$ | 2-layer $w = 7$ | Original | 2-layer $w = 7$ | 1-layer $w = 3$ |

| Original | 1-layer, $w = 3$ | 1-layer, $w = 7$ | 2-layer, $w = 7$ | 2-layer, $w = 7$ and $w = 3$ at $s = 0$. |

Fig. 10 Preserving fine scale details. This figure illustrates that the multi-layer model retrieves geometric structures of the exemplar in a cleaner way. In particular, applying the single-layer model on patches 7×7 is not sufficient to produce relevant structures. Also, one can see that the last 3×3 transport applied at resolution 0 is important to retrieve fine scale structures

time, comparable to methods based on pre-estimated feed-forward networks like [59].

Several synthesis results obtained with the multi-layer model are displayed in Fig. 9. Compared to the previous model [15], it is clear that working on 7×7 patches (instead of 3×3 patches) allows to capture larger geometric structures of the exemplar texture. This is not surprising since the semi-discrete transport maps that are used are essentially weighted NN assignments that use patches seen in the exemplar texture. But still, combining the Gaussian initialization at coarse resolution and the patch averaging procedure at all resolutions permits to create new patches which are not seen in the exemplar. This is where the model gets its capacity of innovation. The copy-paste effect will be discussed below. Let us also mention that the new model is able to better respect fine details, thanks to the last 3×3 transport map applied at the finest resolution, see Fig. 10.

In Fig. 11, we compare the multi-resolution model with other state-of-the-art texture models. The model of Gatys et al. [16] consists in optimizing the image (starting from a white noise) in order to match some spatially averaged second-order responses obtained in a pre-learned neural network. The model of Ulyanov et al. [59] consists in learning a feed-forward neural network that mimics the optimization procedure of Gatys' method. Finally, the model of Raad et al. [46] consists in progressively growing the texture using patch conditional sampling (also working in a coarse-to-fine manner so that one resolution is conditioned by the previous coarser resolution).

One can see that the visual quality is comparable to the one attained by the model of Gatys et al. [16] and is clearly higher than the one obtained by the other models (should it be in terms of details, frequency content, or spectrum). However, compared to [16], the images generated with our model are often smoother; this is probably due to the averaging step used to merge patches at all resolutions. Also, the unstructured texture of the fifth row of Fig. 11 is worth of comment; on this example, both models fail to preserve

the frequency content of the texture, but the failure of our model can be avoided by properly setting the parameter S. Indeed, one should set the number S resolutions depending on the size of structures present in the exemplar image, keeping in mind that the receptive field is of size $2^{S-1}w \times 2^{S-1}w$ (for example, with $S = 4$ resolutions and 7×7 patches, the receptive field is 56×56). For a texture with no salient structure (like the one shown in the 5th row of Fig. 11), taking a single-resolution model (i.e., $S = 1$) is sufficient to obtain a perfect result. Observe that pseudo-periodic patterns are often not well reproduced by neural network methods, while long correlations can be captured by our multiscale model. This phenomenon can be circumvented by adding other terms to the considered loss function, adding Fourier spectrum information [34] or spatial correlation [55].

Additionally, it is interesting to compare the results by looking at the resulting patch distributions, as is proposed in Fig. 12. Since the patch distribution is very high-dimensional, for that, one can plot the one-dimensional distributions obtained after projections in the principal components of the patch space (the principal component analysis being led on the patches of the original texture). Curiously, even if our model is inherently designed to preserve the patch distribution, one can see that the model by Gatys et al. often outperforms it in terms of proximity to the original patch distribution. Again, this may be caused by the averaging step, or also by the fact that the multi-layer maps do not attain exactly the target empirical patch distributions. However, this is very difficult how some observed drifts on the patch distributions (on some principal components) will impact the visual perception. Complementing the study by looking at the color distribution or the spectrum helps to understand the failure/success of the different algorithms.

Let us also remark that, even if our model does not reach the visual quality of Gatys' model, it reaches a very good compromise between visual quality and synthesis time (since, again, the model estimation can be performed offline). In contrast, the model by Ulyanov et al. [59], which can also
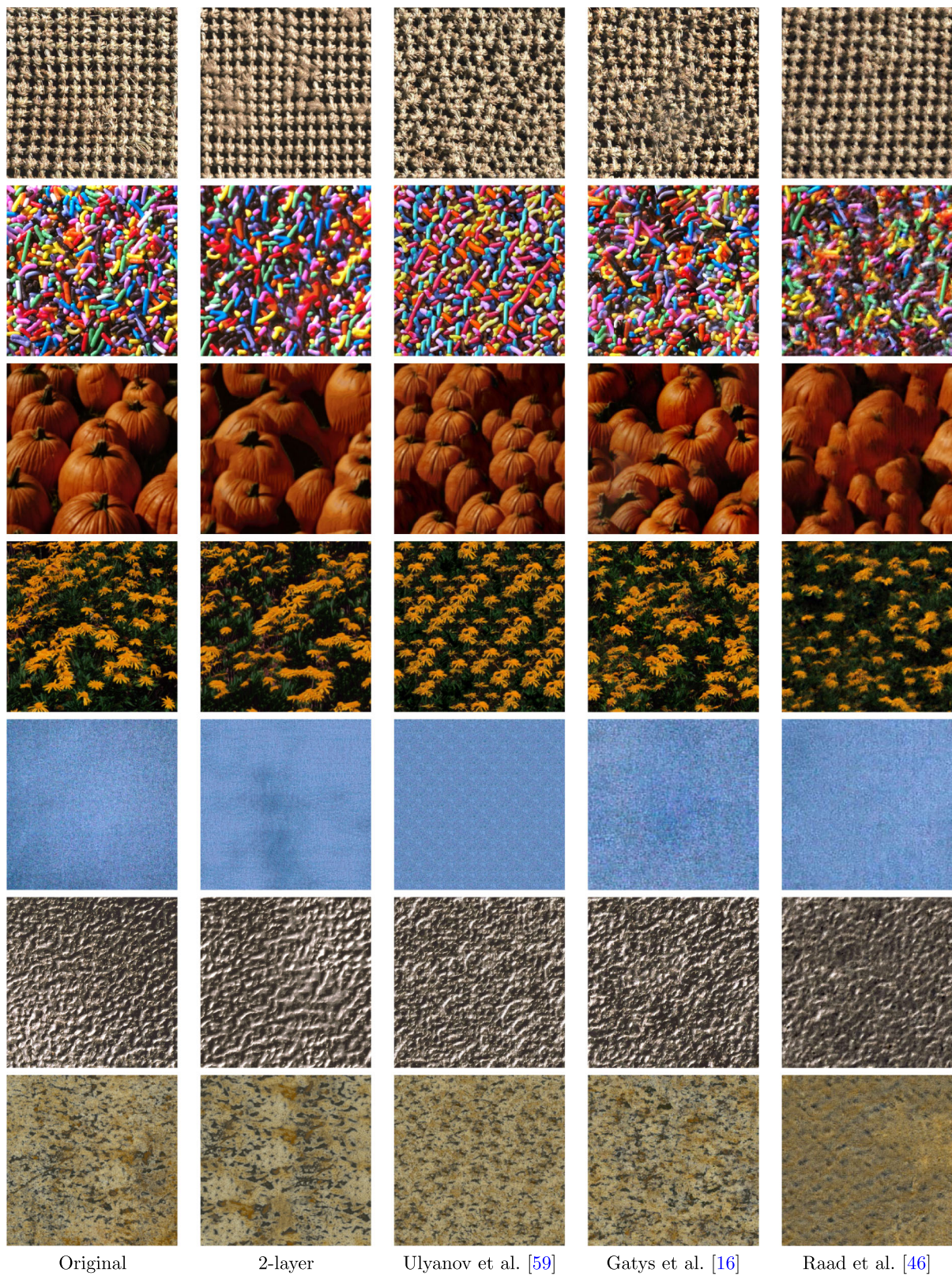
Fig. 11 Comparison. For each row, we display, from left to right, an original texture, the result of the model based on two-layer patch transport, the result of the feed-forward network of [59], the result of the neural optimization procedure of [16] and the results of the patch-based method of [46]. See the text for comments
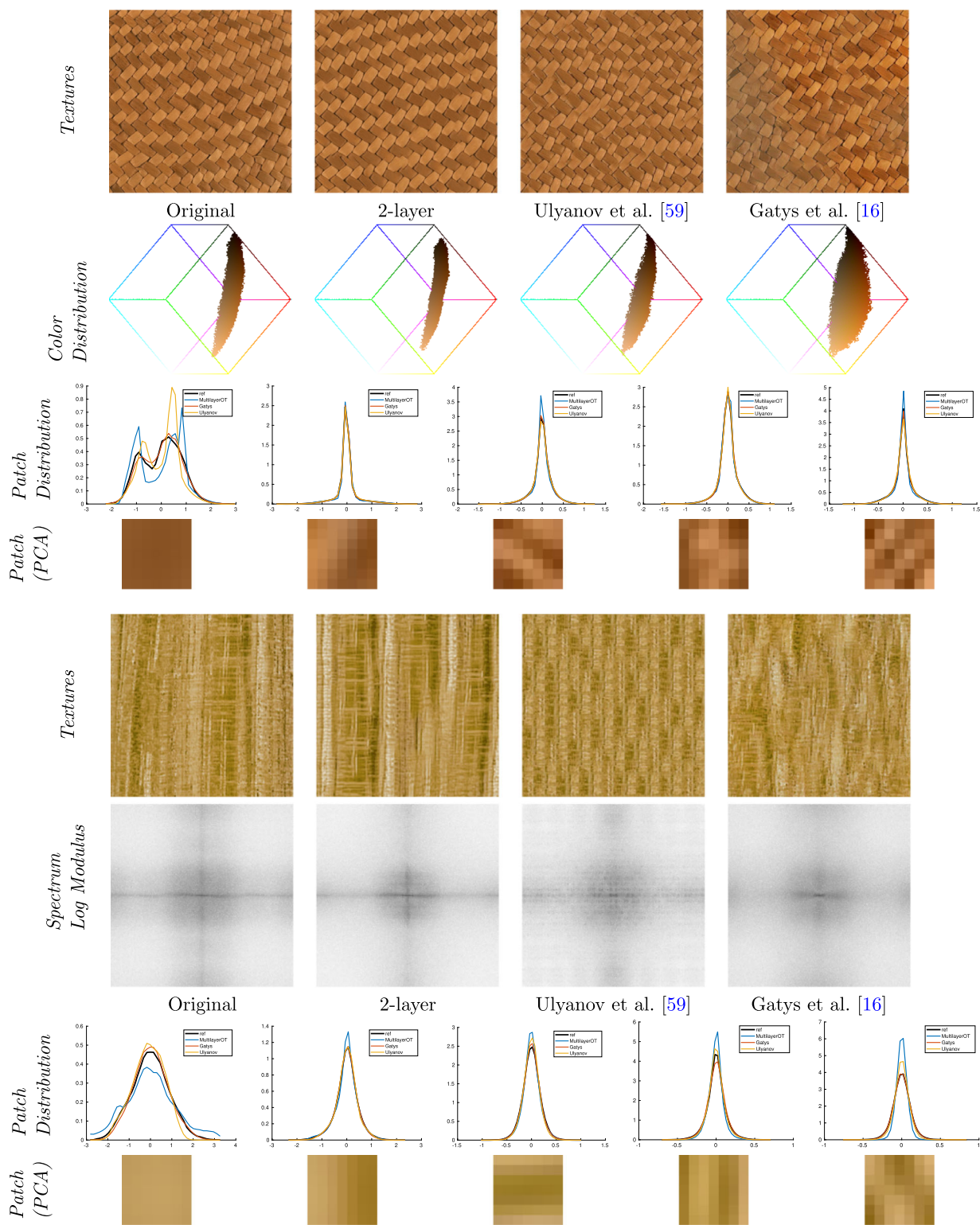
Fig. 12 In this figure, we compare the synthesis results obtained with the model based on bilayer OT and with the models of [59] and [16]. The first and fifth rows contain images synthesized with these three models. In the second row, we display the output color distributions visualized with [33]. The patch distributions obtained with the different models are displayed in the third and seventh rows (summarized by the one-dimensional distribution of the projections on the principal components shown in fourth and eighth rows). In the sixth row, we display the Fourier spectrum of images of the fifth row. See the text for comments
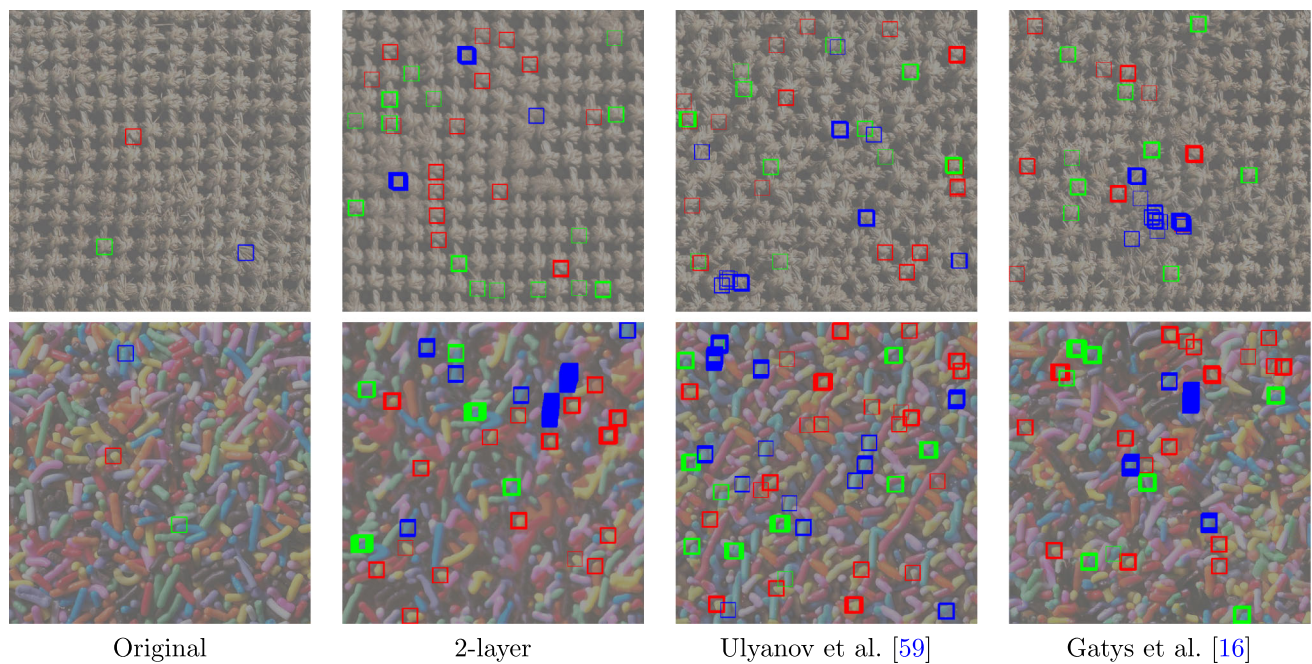
| Original | 2-layer | Ulyanov et al. [59] | Gatys et al. [16] |

Fig. 13 Innovation capacity. In this figure, we illustrate the capacity of innovation of three texture models by looking for several patches of the exemplar texture (marked with red, green and blue squares) the 50 NN in the synthesis (marked with squares with corresponding color). Com- pared to other recent texture synthesis algorithms, our texture model generates patterns which are more similar to those of the exemplar tex- ture (because multi-layer maps projects of patches seen in the exemplar), but always with some local variations due to patch averaging

be estimated offline and has a comparable synthesis time, produces the results with curious visual artifacts that can be analyzed in the patch distribution or the spectrum.

Finally, we propose one last experiment that aims at eval- uating the capacity of innovation of the previous models. For several large patches in the original texture, we propose to single out their NN in the synthesis. The results can be seen in Fig. 13. Since the bilayer transport maps project onto patches seen in the original texture, it is expected that some pieces of the synthesis are quite close to the original. However, one can see that there are always small differences thanks to the patch averaging step.

4.3 Textural Inpainting

To close this section, we propose to show briefly that the texture model defined in the previous section can also be used to address textural inpainting. Indeed, if the original texture $u : \Omega \to \mathbb{R}$ is known outside a mask $M \subset \Omega$, one can define the target patch distribution as the empirical distribution of available patches. Working with several resolutions is not an issue either since one can compute subsampled versions $u_s : \Omega_s \to \mathbb{R}$ and define an adapted mask M_s at resolution s (for example, by thresholding the bilinear reduction in the indicator function of M).

However, at the coarsest scale, the synthesized Gaussian texture should agree with the unmasked content. Fortunately, in the Gaussian case, the inpainting problem can be for- mulated as a conditional Gaussian model estimated outside the mask as proposed in [14]. The benefit of such a condi- tional simulation is that, at the coarsest scale, the synthesized content will extend the long-range correlations that can be observed outside the mask. Then, using this Gaussian inpaint- ing of the coarsest scale, one can apply the bilayer transport maps at several resolutions in order to reimpose geometric structures, as shown in Fig. 14. The model is estimated in the same way than for the case of pure synthesis, except that the target patch distributions contain only patches which do not overlap the mask. (The number of scales should be some- times reduced in order to find enough available patches at the coarsest resolution.) Also, a crucial difference lies in the patch recomposition: One should apply the transport maps to all patches that overlap the mask boundary (and average them to get an image).

Some textural inpainting results can be seen in Figs. 14, 15 and 16. As can be observed in Figs. 14 and 15, this inpainting model allows to inpainting structured textures in a reasonable way. However, for complex sharp textures, some blur can be perceived at the boundary of the mask (for example, in the second example of Fig. 16). Also, the comparison of Fig. 16 highlights the benefit of considering the patch distributions.
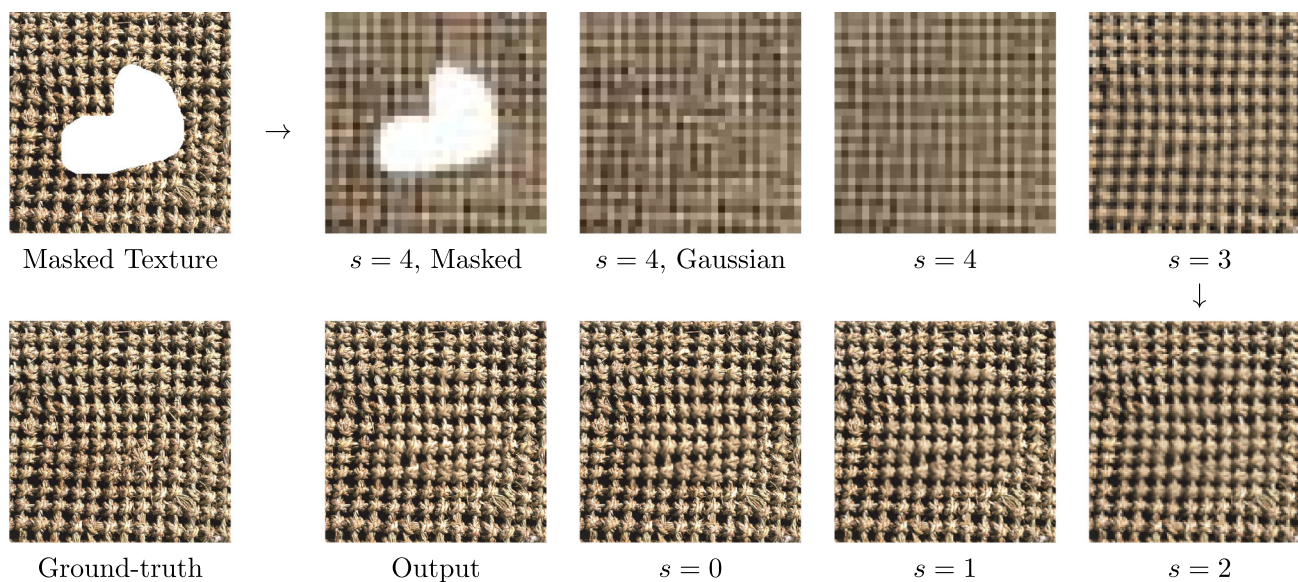
Fig. 14 Multiscale Textural inpainting. The proposed textural inpainting consists in using a Gaussian conditional sampling at the coarsest scale (first row, third column) and then adding details with bilayer patch transport maps

Indeed, constraining the patch distribution allows to avoid inpainting failures that are undesirable optima when optimizing functionals based only on the distance to the patch NN.

Figure 17 compares our approach with two other inpainting techniques [4,61] that have a limited amount of artifacts at the boundary of the mask. On the one hand, *deep image prior* [61] consists in training a fully convolutional neural network (in this specific case, with skipping connections) to generate the masked image from a (fixed) noise input using a mean square error loss function outside the masked domain. Due to the continuous nature of the generative network, this technique gives sometimes surprisingly good results with a seamless transition at the border of the mask. On the other hand, we compare to the greedy patch-based copy method of [4], where an efficient blending technique is proposed to avoid block artifacts when combining locally patches from different locations in the image. Again, our approach compares favorably to those methods, despite introducing some noticeable blur at the finest resolution due to the averaging of patches.

5 Application to Style Transfer

5.1 A Short Review on Image Stylization

The principle of image stylization or *style transfer* (see, e.g., [8,9,13,16,23,24,59,66]) is to give a *source* image the artistic look of an example (or *style*) image, such as a painting, a texture or another picture with the desired visual features

(e.g., color or dynamic range). The most important aspect of this problem is that the synthesized image should at the same time have similar visual features than the style image and preserve the geometrical content of the original image.

Until recently, most successful approaches consisted in patch-based methods that are largely based on texture synthesis [8,23]. Often, the only main difference is that the initialization, instead of using random noise, is the image to be modified itself. However, there is no guarantee that the resulting image will keep most of its important geometrical structures. In order to cope with this issue, other methods rely on a variational formulation that balances the two objectives: An objective function comparing style features to the style image, linearly combined with a second objective comparing other geometric features to the original image. Such formulations are usually used to drive an optimization algorithm that iteratively updates the pixel values of the synthesized image.

As was already discussed in the introduction, textural or style features can be based on patch representations [9,13]. However, a more popular way to extract features is to use a visual representation of images that is learned on a large dataset of natural images, as initiated in [16]. In practice, most techniques use features extracted from layers of the deep convolutional neural network VGG-19 [56] trained on ImageNet, but random representations could be used as well [62]. Note that learned representations for image classification are known to be heavily biased toward textural information [18,39]. In order to use such a network for style transfer, spatial information linked to geometrical features of the input must be kept while the style features must be extracted
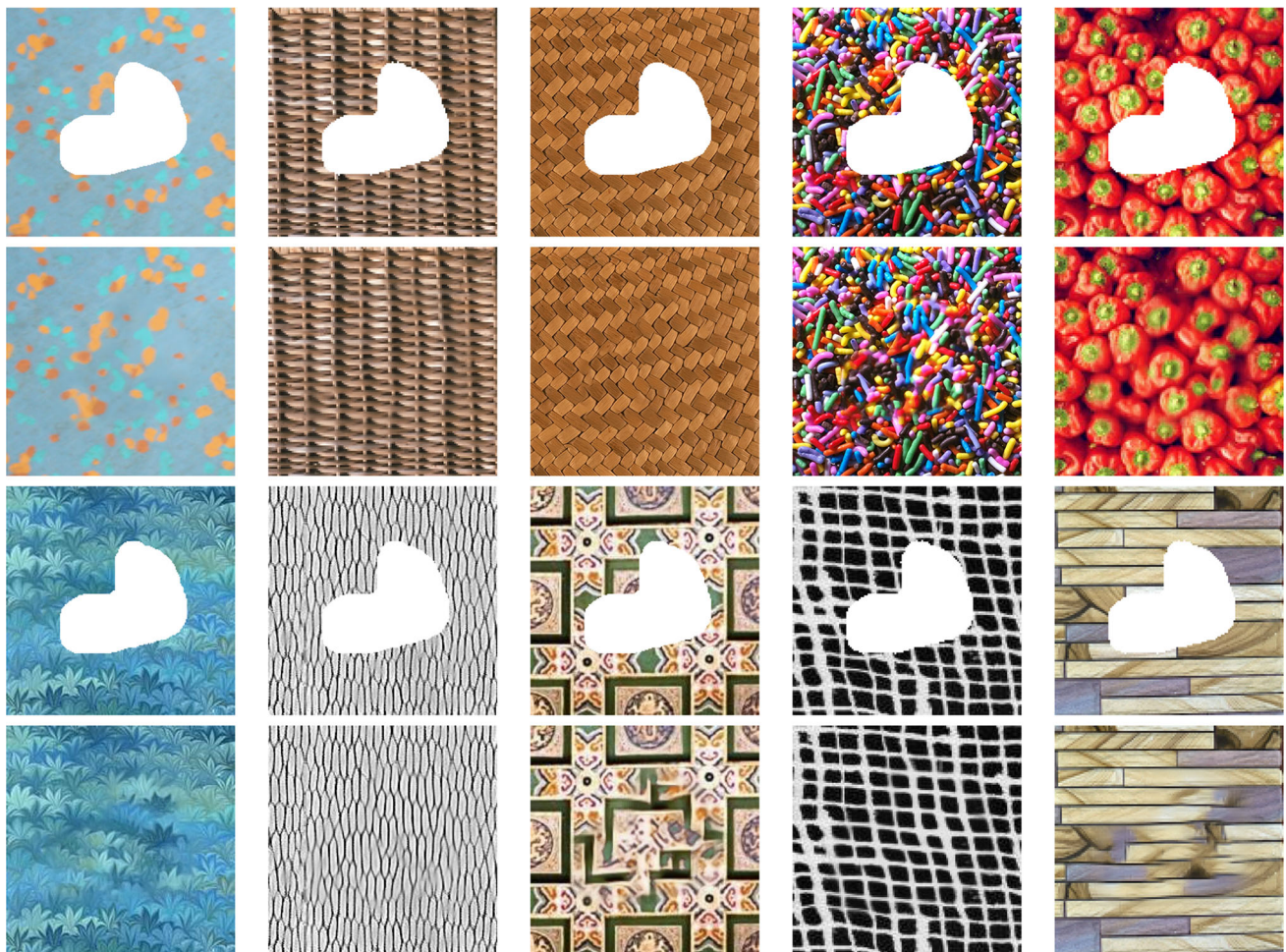
Fig. 15 Textural inpainting results. For several masked textures (first and third rows), we display the inpainting result obtained with the model based on two-layer patch transport. Notice that the inpainted content blends quite nicely with the rest of the image, even if a residual blur is sometimes perceived due to patch averaging

with spatially averaged statistics (often with second-order statistics, like Gram matrices). Although using deep representations makes the optimization more involved (as it requires backward propagation through a deep network), it improves greatly the visual quality [17].

The authors of [24] have shown that this optimization process could be performed offline by training a deep network to achieve a given type of stylization on a dataset (MS-COCO with 80k training images). After training, the corresponding stylization of a new source image is simply done by a single forward propagation through the neural network. Afterward, the authors of [59] demonstrated that such a technique can be used for texture synthesis as well, or for multiple styles [66]. While image quality might not be on par with deep optimization technique [17], the speed-up is of several orders of magnitude, as it requires less computations than for a single iteration which involves forward & backward propagation through a deep descriptive network.

5.2 Feed-Forward Texture Transfer

We propose here to adapt the multi-resolution texture synthesis algorithm of the last section to perform style transfer in a "feed-forward" fashion. More precisely, we estimate the multi-resolution texture model associated with the *style* image, and then, we directly apply the multi-layer transport maps to the *source* image in a coarse-to-fine manner, without any optimization. The main difference is the following: At each scale, the upscaled output from the previous scale is now blended with geometric features from the source image. For simplicity, we use only edges as geometric features. Note that, as for the "perceptual loss" in [16,24,59], style features do not encode spatial information in our case (patch distribution) but geometrical features do. The method is summarized in Algorithm 4 and detailed in the next paragraphs. Contrary to other aforementioned approaches, our algorithm is not optimizing the input image nor trained on a dataset.
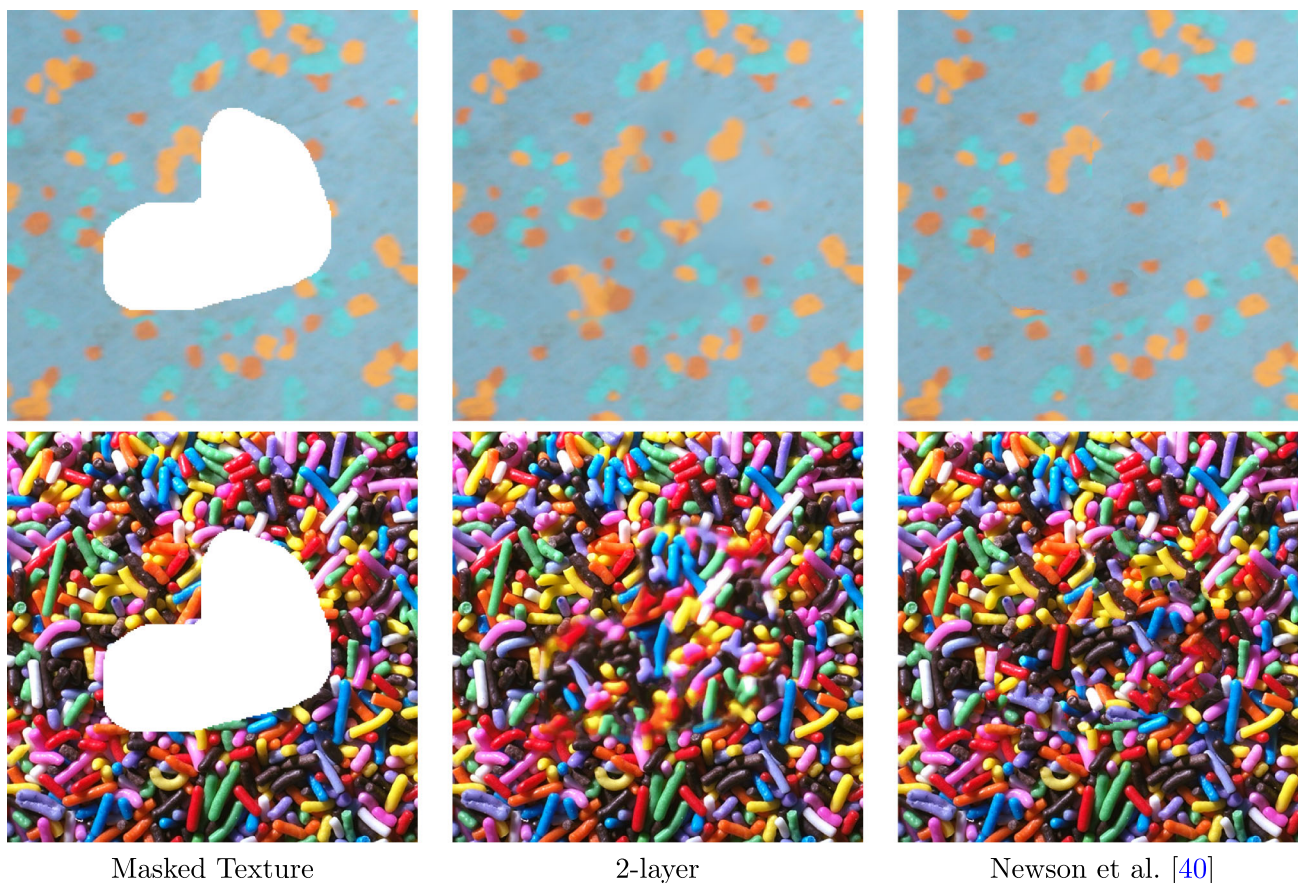
| Masked Texture | 2-layer | Newson et al. [40] |

Fig. 16 Textural inpainting comparison. In this figure, we compare our approach for textural inpainting with the one of Newson et al.[40]. (Their result was obtained with the online demo [41].) Notice that our approach blends the synthesized content with more blur, but also in a way that is less oblivious of the surrounding content. In contrast, with [40], the generated content often disagrees with the context at the boundary of the mask

| Texture | Masked Texture | Our approach | Deep-image prior [61] | Patch blending [4] |

Fig. 17 Textural inpainting comparison. In this example, we compare our approach with *Deep image prior* [61] (which optimizes a deep neural network with skipping connection to generate the masked image using Mean Square Error criterion) and with [4], where patches from the uncovered part of the image are used to fill in the masked domain and blended to avoid blocking artifacts. Notice that the available textural content is better represented with our approach since it is inherently designed to respect the patch distribution
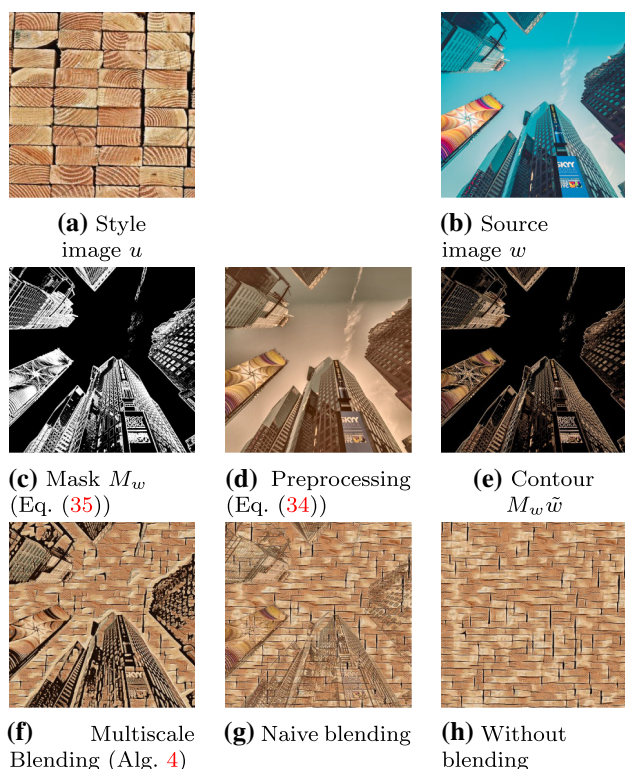
image and normalized using a sigmoid function:

$$M_w(x) = S_t\left(\|\nabla(w * g_\gamma)(x)\|\right) \tag{35}$$

where $*$ indicates the discrete convolution with the Gaussian density function g_γ of standard deviation γ (controlling the detection scale), and

$$S_\tau(w) = \frac{1}{1 + e^{-10\frac{w-\tau}{\tau}}} \tag{36}$$

parameterized by τ, which controls the detection intensity. Figure 18c shows the mask computed at the largest resolution.

The blending of texture synthesis with geometric features is performed with a convex combination between the current upsampled image (33) and the preprocessed image $\tilde{w}_s$ (34) (Fig. 18e)

$$\tilde{U}_s(x) = \left(1 - M_{w_s}(x)\right) U_s(x) + M_{w_s}(x)\tilde{w}_s(x). \tag{37}$$

Once blended, we can apply the pre-learned multi-layer transport map as in (31). Figure 18 shows that applying the blending before the transport map (Fig. 18g) rather than after (Fig. 18h) gives a much better result.

In practice, as shown in Fig. 19, decreasing the threshold τ adds details with less contrast from the input, and decreasing γ keeps only fine details.

Remark Observe that, as we blend the preprocessed input image with textures, the patch distribution of the stylized image does not follow the GMM that served as the source of the transport maps. However, as we change patches only at some edges of the texture image, the difference is practically negligible for this application.

(a) Style image u

(b) Source image w

(c) Mask M_w (Eq. (35))

(d) Preprocessing (Eq. (34))

(e) Contour $M_w\tilde{w}$

(f) Multiscale Blending (Alg. 4)

(g) Naive blending

(h) Without blending

Fig. 18 **Illustration of contour extraction and blending for texture transfer.** This figure contains the style image u (a), the source image w (b), the mask M_w (c) defined from Eq. (35), the preprocessed image $\tilde{w}$ (d) defined from Eq. (34), the contour image $M_w\tilde{w}$ (e), the result of multiscale blending (f) obtained from Alg. 4, the result with naive blending (g), and the synthesis without blending (h). Note that the direct blending of the contours in the synthesized texture is not sufficient (g). Using blending directly during the multi-resolution synthesis (f) gives more satisfying results

Preprocessing We denote by w the input image to be processed and u the style image. In order to ensure that the patch distribution of w at the coarsest resolution $s = S - 1$ is close to the one of u (which serves to estimate the transport maps), we first preprocess w to match the patch second-order statistical properties of u. Namely, the preprocessed image $\tilde{w}$ is constructed by overlapping transformed patches:

$$\tilde{w}(a) = \frac{1}{|\omega|} \sum_{b \in \omega} A(w_{|a-b+\omega})(b), \tag{34}$$

where A is an affine transform that imposes the patch mean and patch covariance of u. Because the reconstructed image $\tilde{w}$ may have out of range values, a local histogram equalization of the intensity channel in Lab color space is performed. An example illustrating this preprocessing is shown in Fig. 18d.

Blending geometric and texture information Geometric features are based on contour extraction. A mask is first computed from the gradient intensity on the smoothed input

---

Algorithm 4: Feed-forward image stylization

1: *Inputs:* style image u and source image w
2: *Offline:* Estimate the transport maps T_s of the texture model defined by u.
3: *Pre-processing:* Compute $\tilde{w}$ (34).
4: *Initialization:* Sample U_{S-1} (29) (same size as w).
5: **for** $s = S - 1, \ldots, 0$ **do**
6:     Resize images at scale s: u_s, w_s and $\tilde{w}_s$
7:     **Contour detection:** Compute mask M_{w_s} (35)
8:     **Blending:** Compute $\tilde{U}_s$ from $\tilde{w}_s$ and U_s (37)
9:     Apply patch transport T_s on patches of $\tilde{U}_s$
10:     Aggregate patches to get V_s (31)
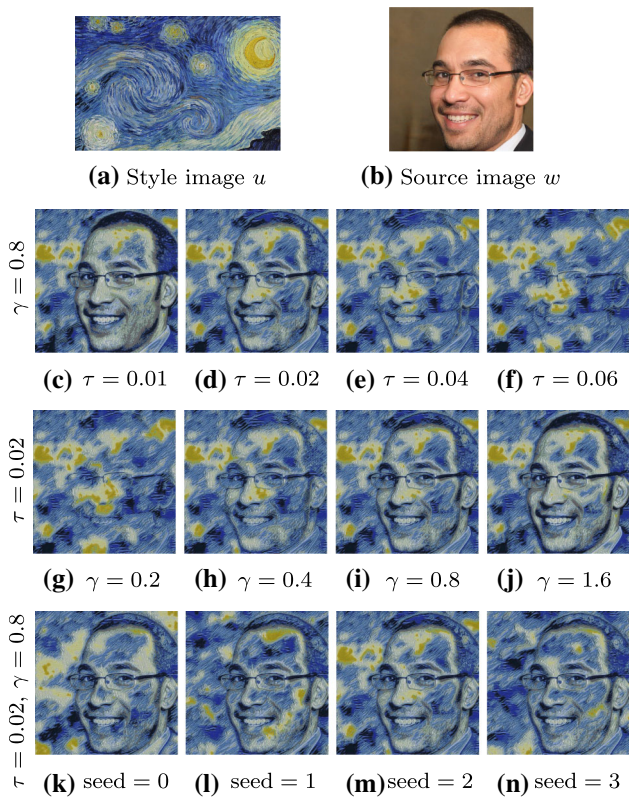11:   If $s > 1$, upsample V_s to get U_{s-1} (33)
12: **return** V_0

(a) Style image u **(b)** Source image w

(c) $\tau = 0.01$ **(d)** $\tau = 0.02$ **(e)** $\tau = 0.04$ **(f)** $\tau = 0.06$

(g) $\gamma = 0.2$ **(h)** $\gamma = 0.4$ **(i)** $\gamma = 0.8$ **(j)** $\gamma = 1.6$

(k) seed $= 0$ **(l)** seed $= 1$ **(m)** seed $= 2$ **(n)** seed $= 3$

Fig. 19 Illustration of contour extraction and blending parameters γ and τ for texture transfer. The same random seed is used to generate the style on the first two lines. The last row shows the effect of changing the random seed used to initialize the algorithm Gaussian random field (29), where $\tau = 0.02$ and $\gamma = 0.8$ (as fixed throughout the rest of experiments)

Additionally, Fig. 19 shows the stylization results with different random initializations. It should be noted that we do not have any of the diversity problems reported for feed-forward networks in [60] and [32] for instance.

5.3 Stylization Results

First, Fig. 20 gives a comparison of our image stylization approach (Fig. 20g) with the gold standard technique of deep image optimization introduced by Gatys et al. [17] (Fig. 20f). As already mentioned, the optimization process is based on an objective function, linearly combining a style loss with a content loss function weighted by a parameter λ. The interesting aspect of using deep image representation is that it enables a more liberal definition of the content of the image to be preserved than ours, resulting in mixing the content of both images in intricate patterns (see, for instance, the posts becoming curved like the branches in the source image). However, as already reported in previous work (see, e.g., [34]), some features are not well preserved, such as color, and checkerboard artifacts may also appear. On the other hand, as previously demonstrated, our technique aims at match-
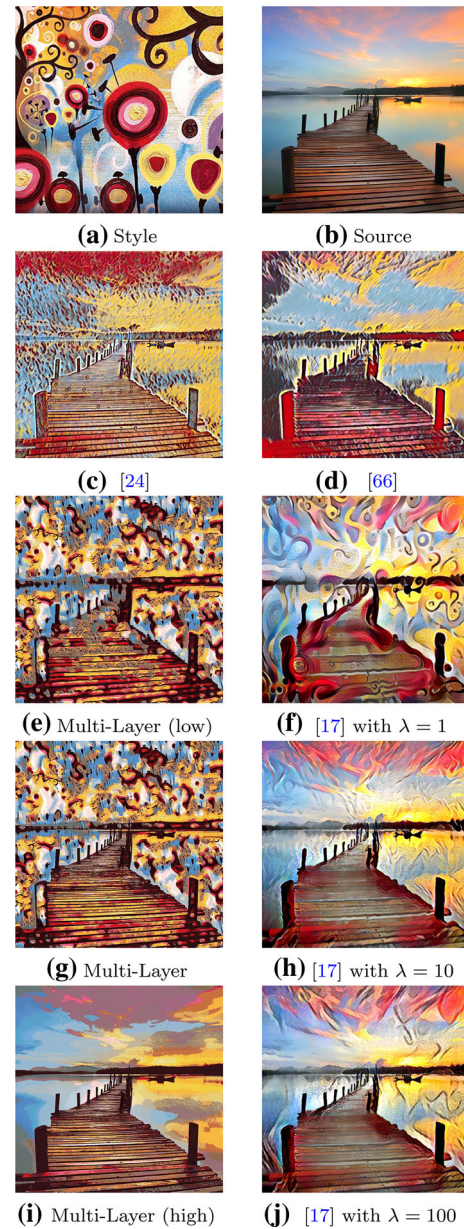

(a) Style   **(b)** Source

(c) [24]   **(d)** [66]

(e) Multi-Layer (low)   **(f)** [17] with $\lambda = 1$

(g) Multi-Layer   **(h)** [17] with $\lambda = 10$

(i) Multi-Layer (high)   **(j)** [17] with $\lambda = 100$

Fig. 20 Image stylization comparison of the proposed approach with other feed-forward techniques requiring learning over a dataset. We display the style image (a), the source image (b), the result of Johnson et al. [24] (c), the result of Zhang et al. [66], the result of our approach for several values of the blending parameter τ (e, g, i), and the result of Gatys et al. [17] for several values of the weighting parameter λ (f, h, j). See the text for comments

ing the patch distribution from the style image, hence being better at preserving color and reproducing tiny details from the exemplar image. For a complete comparison, we display stylization with our method when incorporating only geometric details at low resolution (by setting $\tau = 0$ after the third scale, and referred to as *low* in Fig. 20e) or with all details at high resolution (by setting $\tau = 1$ after the third scale, referred to as *high* in Fig. 20i).
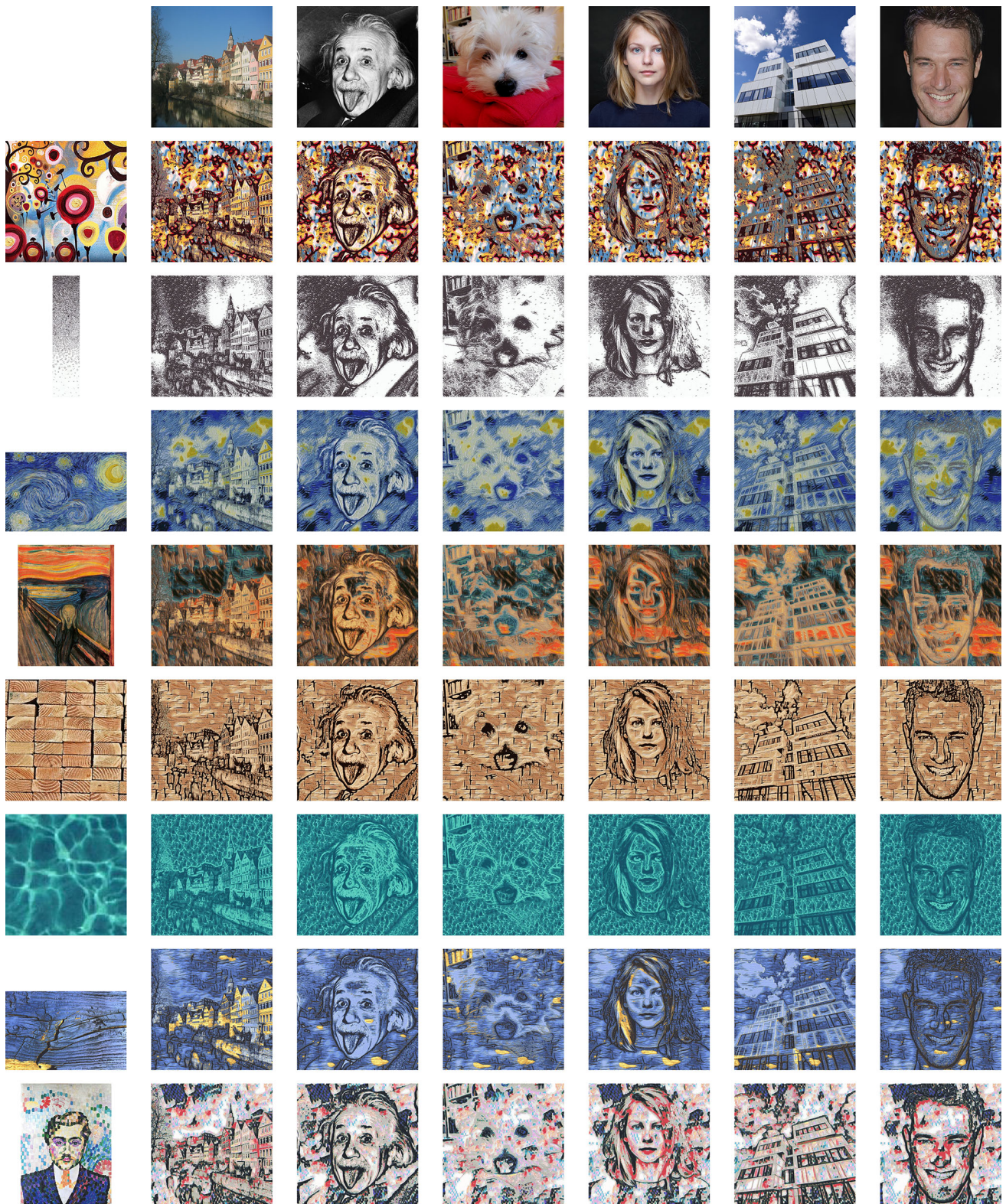
Fig. 21 Feed-forward multi-layer style transfer. Note that the same random seed (i.e., used to generate the Gaussian texture sample at the coarsest scale) is used for all examples, showing how incorporating geometric features in Algorithm 4 changes the outcome. Stylization parameters are fixed throughout all experiments as well ($\gamma = 0.8$ and $\tau = 0.02$) but could be adapted depending on the geometric content of the source image

Comparing our technique to other feed-forward approaches from the literature [24,66] (that still require training on a large dataset of natural images) is interesting as well. Both techniques (Johnson et al. [24] in Fig. 20c, Zhang et al. [66] in Fig. 20d) do not manage to preserve the main features of the style image, thus highlighting the benefit of our statistical framework. Quite interestingly, unlike Gatys et al. [17], these methods have automatically learned to preserve the contours from the source image, as we choose to do explicitly.

Figure 21 shows various stylization results for different types of source image. In these experiments, all stylization parameters are fixed (same random seed, $\tau = 0.02$ and $\gamma = 0.8$) and the same parameters have been used for learning all bilayer transportation maps from the style image (as detailed in Sect. 4.2). To start with, observe that our method is better suited for style images which are textures, which is the main hypothesis of our statistical framework. When it is not the case, we observed that the multi-layer model has more trouble to match statistics with the same set of parameters, requiring more iterations during training, and more sampled patches, more GMM components, etc. Yet, it is still providing interesting results on non-stationary style examples (such as rows 1, 4 and 8 of Fig. 21).

6 Conclusion

In this paper, we have proposed to approximate a semi-discrete OT map with multi-layer transport maps that exploit a hierarchical clustering of the target discrete distribution. After studying the optimality condition for such multi-layer maps, we have shown that they can be optimized with a heuristic stochastic optimization approach. The corresponding algorithm performs better than the usual stochastic algorithm for semi-discrete OT, especially when the target discrete distribution has a very large support. However, we have shown that, except in dimension 1, the hierarchical clustering of the target distribution induces a bias that cannot be coped with by the optimization procedure.

Such multi-layer transport maps, therefore, can be used for applications that rely on a large-scale OT setting (both in terms of dimensions of the underlying space and in terms of support of the target distribution). In particular, we proposed to exploit this multi-layer approach to tackle OT in the patch space, which is useful for several imaging applications, and we gave the results in texture synthesis, texture inpainting and style transfer. For texture synthesis, the multi-layer approach allowed us to consider larger patches in a previous texture model, which considerably enrich the class of well-reproduced textures. We thus obtained a texture model that is comparable to state-of-the-art models in terms of visual quality, while maintaining several mathematical properties (like long-range independence), and a good empirical control on visual statistics (like the color distribution of the spectrum). Besides, once estimated, such a texture model can be sampled in a fast manner (since it consists only in recursive weighted NN patch assignments) and thus has a competitive computational time compared to recent neural network techniques.

This texture model can also be used to address texture inpainting (by relying on a Gaussian conditional sampling scheme for the coarse resolution) or style transfer (with a feed-forward approach that blends the source image with a synthesized texture and which does not need pixel-based optimization). For both these problems, this model produces the visual results that compare well to other competing methods, with the benefit of a good statistical control, in the sense that all the textural content visible in the input texture (or the style image) will be fairly represented in the output image. For inpainting, this statistical control allows to avoid trivial solutions of patch-based approach (for example, flat areas obtained by repeating a flat available patch).

The main limitation of this texture model lies in its constrained architecture. While the number of scales is easily tuned, the patch recomposition strategy is here fixed as a simple average. Although this linear strategy helps to preserve the covariance structure of the random field, it inexorably produces some blur in the synthesized images, which is often the main explication for the failure cases encountered in this article. This issue would certainly be solved by constraining the patch distribution after the patch recomposition strategy. But this would require a radical change of the model architecture that we leave for further investigation.

References

1. Angenent, S., Haker, S., Tannenbaum, A.: Minimizing flows for the Monge–Kantorovich problem. SIAM J. Math. Anal. **35**(1), 61–97 (2003)

2. Aurenhammer, F., Hoffmann, F., Aronov, B.: Minkowski-type theorems and least-squares clustering. Algorithmica **20**(1), 61–76 (1998)

3. Bercu, B., Bigot, J.: Asymptotic distribution and convergence rates of stochastic algorithms for entropic optimal transportation between probability measures. arXiv preprint arXiv:1812.09150 (2019)

4. Buyssens, P., Daisy, M., Tschumperlé, D., Lézoray, O.: Exemplar-based inpainting: technical review and new heuristics for better geometric reconstructions. IEEE Trans. Image Process. **24**(6), 1809–1824 (2015)

5. Carlier, G., Oberman, A., Oudet, E.: Numerical methods for matching for teams and wasserstein barycenters. ESAIM Math. Model. Numer. Anal. **49**(6), 1621–1642 (2015)

6. Cuturi, M.: Sinkhorn distances: Lightspeed computation of optimal transport. In: Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS'13), vol. 2. Curran Associates Inc., Red Hook, NY (2013)

7. Delon, J.: Midway image equalization. J. Math. Imaging Vis. **21**(2), 119–134 (2004)

8. Efros, A., Freeman, W.: Image quilting for texture synthesis and transfer. In: ACM TOG, pp. 341–346 (2001)

9. Elad, M., Milanfar, P.: Style transfer via texture synthesis. IEEE Trans. Image Process. **26**(5), 2338–2351 (2017)

10. Feydy, J., Charlier, B., Vialard, F., Peyré, G.: Optimal transport for diffeomorphic registration. In: International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 291–299. Springer, Berlin (2017)

11. Feydy, J., Roussillon, P., Trouvé, A., Gori, P.: Fast and scalable optimal transport for brain tractograms. In: International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 636–644. Springer, Berlin (2019)

12. Fitschen, J., Laus, F., Steidl, G.: Transport between RGB images motivated by dynamic optimal transport. J. Math. Imaging Vis. **56**(3), 409–429 (2016)

13. Frigo, O., Sabater, N., Delon, J., Hellier, P.: Split and match: example-based adaptive patch sampling for unsupervised style transfer. In: Proceedings of the IEEE CVPR, pp. 553–561 (2016)

14. Galerne, B., Leclaire, A.: Texture inpainting using efficient Gaussian conditional simulation. SIAM J. Imaging Sci. **10**(3), 1446–1474 (2018)

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

16. Gatys, L., Ecker, A.S., Bethge, M.: Texture synthesis using convolutional neural networks. In: Proceedings of NIPS, pp. 262–270 (2015)

17. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2414–2423 (2016)

18. Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F.A., Brendel, W.: Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In: International Conference on Learning Representations (2019)

19. Genevay, A., Cuturi, M., Peyré, G., Bach, F.: Stochastic optimization for large-scale optimal transport. In: Proceedings of NIPS, pp. 3432–3440 (2016)

20. Gerber, S., Maggioni, M.: Multiscale strategies for computing optimal transport. J. Mach. Learn. Res. **18**(1), 2440–2471 (2017)

21. Glimm, T., Henscheid, N.: Iterative scheme for solving optimal transportation problems arising in reflector design. Mathematics (2013) https://doi.org/10.1155/2013/635263

22. Haker, S., Zhu, L., Tannenbaum, A., Angenent, S.: Optimal mass transport for registration and warping. Int. J. Comput. Vis. **60**(3), 225–240 (2004)

23. Hertzmann, A., Jacobs, C.E., Oliver, N., Curless, B., Salesin, D.H.: Image analogies. In: ACM (ed.) Proceedings of SIGGRAPH'01, pp. 327–340. ACM Press, Cambridge (2001)

24. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European Conference on Computer Vision, pp. 694–711. Springer, Berlin (2016)

25. Kitagawa, J.: An iterative scheme for solving the optimal transportation problem. Calc. Var. Partial Differ. Equ. **51**(1–2), 243–263 (2014)

26. Kitagawa, J., Mérigot, Q., Thibert, B.: Convergence of a Newton algorithm for semi-discrete optimal transport. J. Eur. Math. Soc. **21**, 2603–2651 (2019). https://doi.org/10.4171/JEMS/889

27. Kosowsky, J.J., Yuille, A.L.: The invisible hand algorithm: solving the assignment problem with statistical physics. Neural Netw. **7**(3), 477–490 (1994)

28. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Res. Logist. Q. **2**(1–2), 83–97 (1955)

29. Leclaire, A., Rabin, J.: A fast multi-layer approximation to semi-discrete optimal transport. In: Proceedings of SSVM, pp. 341–353. Springer, Berlin (2019)

30. Lellmann, J., Lorenz, D.A., Schönlieb, C., Valkonen, T.: Imaging with Kantorovich–Rubinstein discrepancy. SIAM J. Imaging Sci. **7**(4), 2833–2859 (2014)

31. Lévy, B.: A numerical algorithm for L2 semi-discrete optimal transport in 3D. ESAIM: M2AN **49**(6), 1693–1715 (2015)

32. Li, Y., Fang, C., Yang, J., Wang, Z., Lu, X., Yang, M.H.: Diversified texture synthesis with feed-forward networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3920–3928 (2017)

33. Lisani, J.L., Buades, A., Morel, J.M.: Image color cube dimensional filtering and visualization. Image Process. On Line **1**, 57–69 (2011). https://doi.org/10.5201/ipol.2011.blm-cdf

34. Liu, G., Gousseau, Y., Xia, G.: Texture synthesis through convolutional neural networks and spectrum constraints. In: International Conference on Pattern Recognition (ICPR), pp. 3234–3239. IEEE (2016)

35. Maas, J., Rumpf, M., Schönlieb, C., Simon, S.: A generalized model for optimal transport of images including dissipation and density modulation. ESAIM Math. Model. Numer. Anal. **49**(6), 1745–1769 (2015)

36. McLachlan, G., Krishnan, T.: The EM Algorithm and Extensions, vol. 382. Wiley, New York (2007)

37. Mérigot, Q.: A multiscale approach to optimal transport. Comput. Graph. Forum **30**(5), 1583–1592 (2011)

38. Monge, G.: Mémoire sur la théorie des déblais et des remblais. Histoire de l'Académie Royale des Sciences de Paris (1781)

39. Mordvintsev, A., Pezzotti, N., Schubert, L., Olah, C.: Differentiable image parameterizations. Distill (2018). https://doi.org/10.23915/distill.00012

40. Newson, A., Almansa, A., Fradet, M., Gousseau, Y., Pérez, P.: Video inpainting of complex scenes. SIAM J. Imaging Sci. **7**(4), 1993–2019 (2014). https://doi.org/10.1137/140954933

41. Newson, A., Almansa, A., Gousseau, Y., Pérez, P.: Non-local patch-based image inpainting. Image Process. On Line **7**, 373–385 (2017). https://doi.org/10.5201/ipol.2017.189

42. Oberman, A.M., Ruan, Y.: An efficient linear programming method for optimal transportation. arXiv preprint arXiv:1509.03668 (2015)

43. Papadakis, N., Peyré, G., Oudet, E.: Optimal transport with proximal splitting. SIAM J. Imaging Sci. **7**(1), 212–238 (2014)

44. Papadakis, N., Provenzi, E., Caselles, V.: A variational model for histogram transfer of color images. IEEE Trans. Image Process. **20**(6), 1682–1695 (2010)

45. Peyré, G., Cuturi, M.: Computational optimal transport. Found. Trends Mach. Learn. **11**(5–6), 355–607 (2019)

46. Raad, L., Desolneux, A., Morel, J.: A conditional multiscale locally Gaussian texture synthesis algorithm. J. Math. Imaging Vis. **56**(2), 260–279 (2016)

47. Rabin, J., Delon, J., Gousseau, Y.: A statistical approach to the matching of local features. SIAM J. Imaging Sci. **2**(3), 931–958 (2009)

48. Rabin, J., Delon, J., Gousseau, Y.: Removing artefacts from color and contrast modifications. IEEE Trans. Image Process. **20**(11), 3073–3085 (2011)

49. Rabin, J., Papadakis, N.: Convex color image segmentation with optimal transport distances. In: International Conference on Scale

Space and Variational Methods in Computer Vision, pp. 256–269. Springer, Berlin (2015)

50. Rangarajan, A., Chui, H., Bookstein, F.L.: The softassign procrustes matching algorithm. In: Biennial International Conference on Information Processing in Medical Imaging, pp. 29–42. Springer, Berlin (1997)

51. Rubner, Y., Tomasi, C., Guibas, L.: The earth mover's distance as a metric for image retrieval. Int. J. Comput. Vis. **40**(2), 99–121 (2000)

52. Santambrogio, F.: Optimal Transport for Applied Mathematicians. Birkäuser, New York (2015)

53. Schmitzer, B.: A sparse multiscale algorithm for dense optimal transport. J. Math. Imaging Vis. **56**(2), 238–259 (2016)

54. Schmitzer, B.: Stabilized sparse scaling algorithms for entropy regularized transport problems. SIAM J. Sci. Comput. **41**(3), A1443–A1481 (2019)

55. Sendik, O., Cohen-Or, D.: Deep correlations for texture synthesis. ACM Trans. Graph. **36**(5), 161:1–161:15 (2017). https://doi.org/10.1145/3015461

56. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

57. Solomon, J., De Goes, F., Peyré, G., Cuturi, M., Butscher, A., Nguyen, A., Du, T., Guibas, L.: Convolutional wasserstein distances: efficient optimal transportation on geometric domains. ACM Trans. Graph. (TOG) **34**(4), 66 (2015)

58. Tartavel, G., Peyré, G., Gousseau, Y.: Wasserstein loss for image synthesis and restoration. SIAM J. Imaging Sci. **9**(4), 1726–1755 (2016)

59. Ulyanov, D., Lebedev, V., Vedaldi, A., Lempitsky, V.: Texture networks: feed-forward synthesis of textures and stylized images. In: Proceedings of the International Conference on Machine Learning, vol. 48, pp. 1349–1357 (2016)

60. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6924–6932 (2017)

61. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Deep image prior. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 9446–9454 (2018)

62. Ustyuzhaninov, I., Brendel, W., Gatys, L., M., B.: What does it take to generate natural textures? In: Proceedings of ICLR (2017)

63. Villani, C.: Topics in Optimal Transportation. American Mathematical Society, Providence (2003)

64. Xia, G., Ferradans, S., Peyré, G., Aujol, J.: Synthesizing and mixing stationary Gaussian texture models. SIAM J. Imaging Sci. **7**(1), 476–508 (2014)

65. Yu, G., Sapiro, G., Mallat, S.: Solving inverse problems with piecewise linear estimators: from Gaussian mixture models to structured sparsity. IEEE Trans. Image Process. **21**(5), 2481–2499 (2012)

66. Zhang, H., Dana, K.: Multi-style generative network for real-time transfer. arXiv preprint arXiv:1703.06953 (2017)

Arthur Leclaire is an associate Professor in Applied Mathematics at the University of Bordeaux (France). He studied mathematics and image processing at ENS Paris and ENS Cachan, and received the Ph.D. from Paris Descartes University in 2015. He held a junior teacher-researcher position at ENS Cachan from 2014 to 2018. He is a specialist of stochastic texture modelling, and his research interests include Fourier analysis for imaging, image restoration, and optimal transport.

Julien Rabin graduated from École Normale Supérieure de Cachan, France, in 2006. He received his Ph.D. in image processing and computer vision from Télécom ParisTech in 2009, under the supervision of Julie Delon and Yann Gousseau. Since 2011, Julien is an Associate Professor at GREYC, an image processing laboratory at ENSICAEN and Normandie University.