Paper Study - A Texture Synthesis Model Based on Semi-Discrete Optimal Transport in Patch Space

Clément Weinreich and Paul-Henri Pinart
Master MVA

January 13, 2024

Abstract

Texture synthesis is a technique that consists in generating some realistic textures by imitating the patterns of a given exemplar texture. It plays a crucial role in many domains such as computer graphics, where the synthesis quality directly impacts the visual quality and the realism of graphical environments. The objective is to have a realistic replica in a visual sense but with original content: we don't simply want to copy parts of the original texture for the sake of visual quality and realism. In this paper review, we will study the work of B. Galerne, A. Leclaire and J. Rabin [4] which uses a patched-based method. The main novelty of the method is based on the use of semi-discrete optimal transport - ie between a continuous and a discrete distribution- to reimpose the patch distribution of the exemplar texture. The goal of this project is to provide quantitative measures of the statistical properties of patches from the synthesis, and visually compare synthesis results with and without constraints. We will also propose some extensions of the method and critically discuss their effectiveness and results.

1 Introduction

One can define texture synthesis as the ability to create realistic and visually pleasing original textures based on an exemplar texture. A texture itself is an image that contains repetitions or patterns, therefore giving some sense to the possibility of its synthesis. Texture synthesis aims to generate some new content that should not be a copy of parts of the original texture but rather an extension that captures well its visual characteristics and in a mathematical context, its statistical properties.

Texture synthesis plays a very important role in many domains like computer graphics, computer vision, and image processing. In computer graphics, it is used to improve the realism of virtual environments and provide a more immersive experience. One could for example think about the creation of a virtual house where we would want a wooden texture for the walls and house tiles for the roof. In computer vision, texture synthesis facilitates tasks such as object recognition and segmentation by providing diverse and representative training data. Furthermore, in image processing, it finds applications in tasks such as image editing, inpainting, and denoising.

There exist different methods in the literature for such a task, but one of the main approaches for texture synthesis is patch-based texture synthesis. Patches are small squares of pixels of an image of fixed size, for example, 3 pixels by 3 pixels patches. This technique involves breaking down the texture into smaller patches, and then synthesizing the target texture by wisely re-assembling these patches. The synthesis is mainly guided by the statistical properties of the input texture, ensuring that the generated texture has similar characteristics. This approach works reasonably well, as conjectured by Julesz in the 1980s [6]. In this article, Julesz introduces the term "textons" to represent basic visual elements that contribute to our perception of textures. Later, in 2000, Portilla and Simoncelli introduced a set of 710 statistical measures [7] sufficient to characterize a very large number of textures. This brought a solid foundation for more advanced texture synthesis and analysis methods.

The successful results of the method of Galerne [4] are based, among other things, on two major components: multiscale optimization and semi-discrete optimal transport. The first consists in studying the texture at different scales (eg 4, 16, 64,... times smaller) and ensuring that the statistical characteristics are kept at each scale between the original texture and the synthesis. Then, semi-discrete optimal transport is used to efficiently transport and redistribute sampled patches from a continuous source distribution to the discrete target distribution. It benefits from a solid mathematical framework and ensures that the synthesized texture adheres closely to the distribution of patches in the exemplar texture.

The structure of this study is designed to provide a comprehensive understanding of the subject. We begin by providing background information on the main mathematical areas essential to the topic. This information is intended to facilitate the understanding of the method, which is described in detail in the following section. Next, we propose and discuss some extensions to the method, and give a critical overview. The study concludes with a presentation of numerical and graphical results, to provide additional support for the concepts discussed.

2 Background

2.1 Semi-discrete optimal transport formulation

Optimal transport is the mathematical study of sending masses of one probability distribution to another in an optimal way, ie by minimizing a certain transportation cost function. The notion of semi-discrete optimal transport refers to the case where we compute the optimal transport between a continuous probability measure and a discrete probability measure. More specifically, given a continuous measure μ and a discrete measure $\nu = \sum_{y \in S} \nu_y \delta_y$ (S finite set) on a metric space $\mathcal{X} = \mathbb{R}^d$, the goal is to find an optimal transport map that minimizes the total cost of moving mass from μ to ν. We first define the push-forward measure which can be seen as a transport map in our case.

Definition 1 (Push-forward measure)**.** *The push-forward measure of a probability measure μ through a measurable map $T : \mathcal{X} \to \mathcal{Y}$, denoted $T_{\#}\mu$, is defined by*

$$(T_{\#}\mu)(A) = \mu(T^{-1}(A)) \tag{1}$$

for any measurable set $A \subset \mathcal{Y}$.

Monge's initial problem formulation is based on push-forward measures, but the optimal transport problem can be formulated using the Kantorovich formulation. This allows a convex relaxation of the problem, which is expressed as follows:

$$\min_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{X} \times \mathcal{X}} c(x,y)\, d\pi(x,y) \tag{2}$$

Here, $\Pi(\mu,\nu)$ is the set of probability measures on $\mathcal{X} \times \mathcal{X}$ with marginal distributions μ, ν, $c(x,y)$ is the cost function representing the transportation cost between x and y, and π is a transport plan. One advantage of considering semi-discrete optimal transport is that the formulation above can be simplified using its convex dual.

Proposition 1. *Let's assume as above that μ is continuous and $\nu = \sum_{y \in S} \nu_y \delta_y$ is discrete. Then solving Kantorovitch's formulation (2) is equivalent to considering biased nearest neighbor assignments, namely :*

$$T_v(x) = \arg\min_{y \in S} c(x,y) - v(y) \tag{3}$$

Therefore, to find an optimal map T_v, we now have to find a "good" v. Once it is found, then transport can be applied efficiently, with patches in our situation.

Remark 1. *We can visualize this semi-discrete optimal transport when $v = 0$ in 2D with Voronoi cells as shown in Figure 1. The dots correspond to the discrete measure dirac locations and the space to the continuous measure. In our case, we do not have a simple nearest neighbor assignment in the sense that the cost function c is biased with v.*

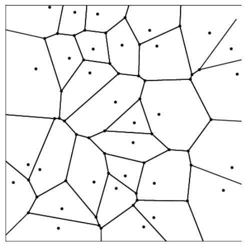

Figure 1: Voronoi cells on a 2D surface.

Remark 2. *In our problem, we do not have Voronoi cells anymore, but instead a power diagram (also called Laguerre–Voronoi diagram or radical Voronoi tesselation). It generalizes the case where the nearest neighbor assignment is biased. We define it in the following.*

Definition 2. *Suppose we have a map T_v as defined in eq. (3). Then its preimages define a power diagram, which is a partition of $\mathbb{R}^D$. For each point $y \in \mathbb{R}^D$, it is defined as :*

$$Pow_v(y) = \{x \in R^D | \forall z \in \{S\}, ||x-y||^2 - v(y) < ||x-z||^2 - v(z)\}. \tag{4}$$

From now on, we will consider that the cost c is of the form $c(x, y) = ||x-y||^2$, although most results are compatible with sufficiently smooth convex cost functions. For the next section, we also define the c-transform v^c as $v^c(x) = \min_{y \in S} c(x, y) - v(y)$.

2.2 Approximation of optimal transport maps

In the previous part, we have reduced the space of acceptable solutions of the semi-discrete optimal transport to a biased nearest neighbor assignment defined in eq. (3). However, it is not yet optimal. We need to optimize the bias v to minimize the transportation cost. This can be done using the following theorem, which introduces a function H to maximize. We will then be able to apply gradient-based methods to obtain an approximation of the optimal transport map.

Theorem 1. *The solutions of the semi-discrete optimal problem defined in eq. (2) are of the form T_v (defined in eq. (3)), where v solves the concave optimization problem :*

$$\arg\max_{v \in \mathbb{R}^S} H(v) \quad \textit{where} \quad H(v) = \int_{\mathbb{R}^D} v^c(x) d\mu(x) + \sum_{y \in S} v(y)\nu_y \tag{5}$$

Furthermore, H is $\mathcal{C}^1$ smooth and its gradient with respect to $v(y)$ is given by

$$\frac{\partial H}{\partial v(y)} = -\int_{Pow_v(y)} \rho(x)dx + \nu_y = -\mu(Pow_v(y)) + \nu_y \tag{6}$$

The formula above gives interesting features. In particular, we obtain a critical point with a relatively simple closed form: $\nu_y = \mu(Pow_v(y))$. However, the main resulting issue is the high computational cost. Calculating the μ-measures of each power cell is expensive, especially in high dimensions. Therefore, stochastic optimization can be a more relevant alternative. The original paper even uses ASGD (Average Stochastic Gradient Descent), which is a variation of SGD that reduces variance by maintaining an average of past terms. This results in a smoother convergence with potential regularization effects, allowing for a more stable optimization process. Furthermore, the additional memory usage and computational complexity compared to SGD remain reasonable if one does not store all previous samples.

Recall the definition of H given in eq. (5), then we define $h : (x, v) \mapsto v^c(x) + \sum\limits_{y \in S} v(y)\nu_y$ which is the density of H w.r.t. μ : $H(v) = \mathbb{E}[h(X, v)]$. Recalling the expression of the c-transform, the gradient of h w.r.t. v is easy to compute :

$$\nabla_v h(x, w) = -e_{T_w(x)} + v$$

where where (e_y) is the canonical basis of $\mathbb{R}^S$

Definition 3 (ASGD, [5])**.** *Average Stochastic Gradient Descent is a variation of SGD that computes the average of the past elements obtained by SGD. Initializing at $\tilde{v}^1 = 0$, we define the terms of the sequence recursively :*

$$\begin{cases} z^k = z^{k-1} + \frac{C}{\sqrt{k}} \nabla_z h(x^k, z^{k-1}) \quad \text{where } x^k \sim \mu \\ v^k = \frac{1}{k}(v^1 + \ldots + v^k). \end{cases} \tag{7}$$

In this review, we will not delve into the convergence proof which is detailed in the original paper. Instead, we will focus in the next part on the method, and then deepen our analysis by proposing extensions and experiments.

3 Patch-based semi-discrete optimal transport for texture synthesis

This section presents the stochastic texture synthesis model proposed by the authors, which is based on transforming a Gaussian random field locally. Initially, the Gaussian random field is responsible for the capture of the texture's long-range correlations, excluding its structured geometric characteristics. Subsequently, the model reintroduces geometric structures through a patch-based transformation. This transformation is designed to be an approximate solution to a semi-discrete optimal transport problem, which ensures statistical alignment with the exemplar texture. Finally, a multi-scale framework enables the efficient synthesis of structured textures while maintaining statistical guarantees. This section presents the framework in two steps. To begin with, we focus on the the first level of synthesis (coarse scale), and then on the synthesis of the other scales which enables refinement and proper respect of the exemplar texture's structure.

3.1 Synthesis at coarse scale

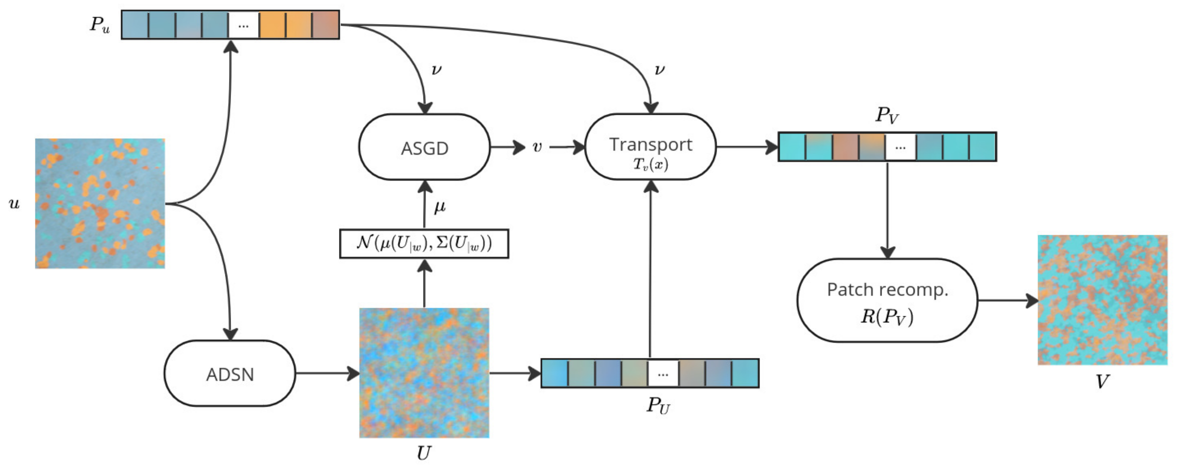

Figure 2: **mono-scale synthesis model**: A source texton u is transformed with ADSN (Asymptotic Discrete Spot Noise) to create a stationary Gaussian random field U that can be used as a continuous distribution μ. A semi-discrete optimal transport problem between the source distribution μ and the target discrete distribution ν (which corresponds to the empirical patch distribution of u) is solved by computing v thanks to ASGD (Average Stochastic Gradient Descent). The optimal map T_v defined in equation 3 is then applied to P_U which gives the transformed patches P_V, that can finally be recomposed into an image using the operator R.

The first step of this synthesis can also be seen as a mono-scale version of the texture synthesis model and is summarized in Figure 2. Let's denote $u : \Omega \to \mathbb{R}^d$ the exemplar texture defined on a domain $\Omega \subset \mathbb{Z}^2$. The first step consists of modeling u as a stationary Gaussian random field U whose first and second order moments are the empirical mean and covariance of the exemplar texture. For this purpose, the Asymptotic Discrete Spot Noise (ADSN) is used ([2], [3]):

$$\forall x \in \mathbb{Z}^2, \quad U(x) = \bar{u} + \sum_{y \in \mathbb{Z}^2} t_u(y) W(x-y) \quad \text{where} \begin{cases} \bar{u} = \frac{1}{|\Omega|} \sum u(x), \\ t_u = \frac{1}{\sqrt{|\Omega|}} (u - \bar{u}) \mathbb{1}_\Omega \end{cases} \tag{8}$$

where $\bar{u}$ is the empirical mean of u and W is a normalized Gaussian white noise on $\mathbb{Z}^2$. One can easily see that $\mathbb{E}[U] = \bar{u}$.

Proposition 2. *The expectation of U is equal to the empirical mean of u, i.e., $\mathbb{E}[U] = \bar{u}$, and the covariance of U corresponds to the autocorrelation function:*

$$Cov(U(x), U(y)) = a_u(y-x) \quad \text{with } a_u(x) = \sum_{z \in \mathbb{Z}^2} t_u(z) t_u(x+z)^T.$$

Proof. The computation $\mathbb{E}[U] = \bar{u}$ is direct from the definition of U. For the covariance, let $x, y \in \Omega$ we have

$$\begin{aligned} Cov(U(x), U(y)) &= \mathbb{E}\left[(U(x) - \bar{u})(U(y) - \bar{u})^T\right] \\ &= \mathbb{E}\left[\sum_{z_1 \in \mathbb{Z}^2} \sum_{z_2 \in \mathbb{Z}^2} t_u(z_1) W(x - z_1) W(y - z_2)^T t_u(z_2)^T\right] \\ &= \sum_{z \in \mathbb{Z}^2} t_u(z) t_u(y - x + z)^T \quad \text{since } \mathbb{E}[W(x-z_1)W(y-z_2)^T] = \mathbb{1}_{\{x - z_1 = y - z_2\}} \\ &= a_u(y-x) \end{aligned} \tag{9}$$

□

The random field U has the correct correlations but no salient structures. To reimpose the statistics of the exemplar texture on local features, a local transform $T : \mathbb{R}^{dw^2} \to \mathbb{R}^{dw^2}$ is applied in the patch space $\mathbb{R}^{dw^2}$ where $w \in \mathbb{N}^*$ corresponds to the patch size. This framework uses an optimal transport map between the distribution of the Gaussian patches of U and the empirical patch distribution of the exemplar texture is used. The source distribution μ of this transport is $U_{|\omega}$, where $\omega = \{0, \ldots, w-1\}^2$ denotes the patch domain. As U is a stationary Gaussian random field, μ corresponds to the distribution of any Gaussian patch of U. Moreover, as the expectation and covariance of U are known, the parameters of the Gaussian distribution μ can be computed, which enables sampling the distribution $\mathcal{N}(\mu(U_{|w}), \Sigma(U_{|w}))$.

For the target measure of the transport, we consider the empirical distribution of patches in the exemplar texture u. Let's note $P_u = \{u_{|x+\omega} | x + \omega \subset \Omega\}$ the set of patches of u, the distribution ν_{emp} is

$$\nu_{emp} = \frac{1}{|P_u|} \sum_{p \in P_u} \delta_p. \tag{10}$$

As texture images contain thousands of patches, it is impractical for ASGD-based optimal transport regarding the computational cost. Thus, ν_{emp} is approximated by the subsampled distribution

$$\nu = \frac{1}{J} \sum_{j=1}^{J} \delta_{p_j} \tag{11}$$

where $p_1, \ldots, p_J$ are $J = 1000$ patches drawn uniformly from P_u. These samples are collectively denoted by the set Y. When $|P_u| < 1000$ there is no need to subsample so $\nu = \nu_{emp}$. Finally, μ and ν are two probability measures on $\mathbb{R}^{dw^2}$. The semi-discrete optimal transport from μ to ν can be computed using ASGD as explained in section 2.2. The optimal set of weights v computed from ASGD can then be used to compute the optimal patch assignment $T = T_v$. In this context, the weighted nearest neighbor assignment is defined by

$$T_v(p) = \underset{(p_j)_{j=1,\ldots,J}}{\arg\min} \; ||p - p_j||^2 - v_j. \tag{12}$$

The mapping T_v is applied to P_U, the set of patches of the synthesis U. This gives a new set of transported patches P_V. To form the final synthesis V, the patches of P_V are recomposed using a recomposition operator that we define as $R : \mathbb{R}^{N\times dw^2} \to \mathbb{R}^{H\times W\times d}$. In this framework, the recomposition operator corresponds to a simple averaging of the overlapping patches.

Definition 4. *We note $\mathcal{V}(x)$ ($x \in \mathbb{Z}^2$) the set of patches that overlap a certain pixel at position x and p_x the value at position x of the pixel in the overlapping patch. Then the averaging operation is defined as:*

$$V(x) = R(P_V)(x) = \sum_{p\in\mathcal{V}(x)} \frac{1}{|\mathcal{V}(x)|} p_x. \tag{13}$$

3.2 Synthesis at finer levels

The mono-scale model can be extended to a multi-scale model. The approach begins with Gaussian synthesis at coarse scale as explained in section 3.1, and then iteratively applies local transformations to reestablish the patch distributions across finer scales. To transition from one scale to another, a simple upsampling method is applied: the upsampled patches correspond to the patches at the same position of the previous scales, but twice larger.

Let us note u^l the subsampled version of the original image at scales $l = 0, \dots, L-1$, and ν^l the approximation of the empirical patch distribution at scale l with associated samples Y^l. The subsampled versions of u are obtained by successive bicubic subsampling of $u = u^0$ by a factor of 2. The previous model presented in section 3.1 considers the synthesis at the coarsest scale $l = L-1$. For finer scales, a slightly different approach is adopted and illustrated in Figure 3.

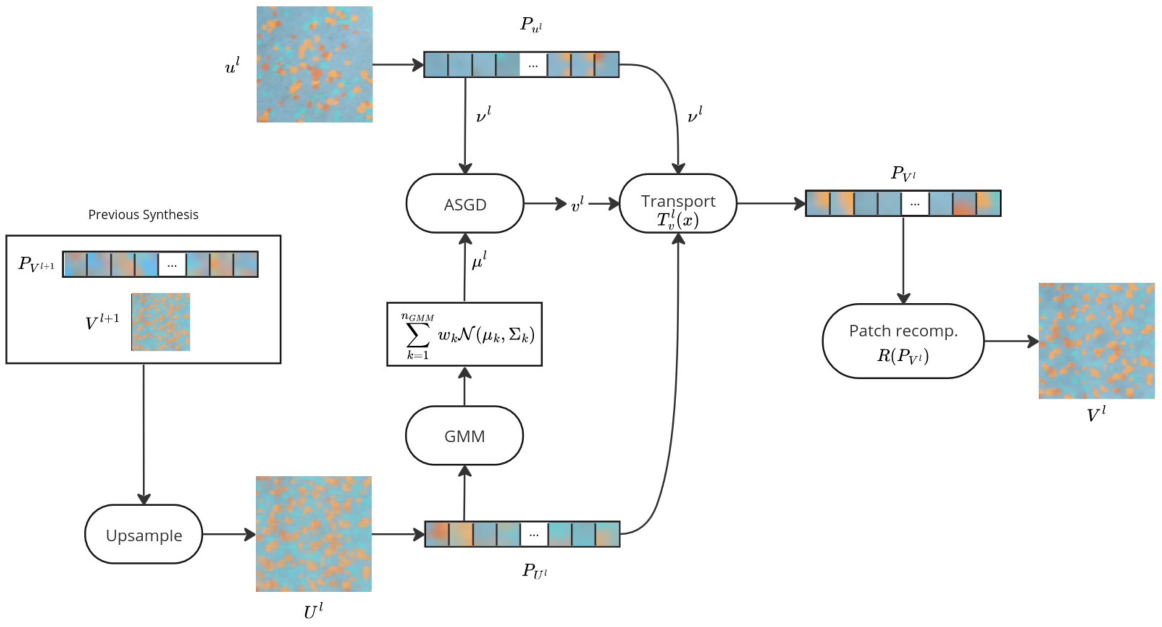

Figure 3: **Synthesis at finer scales**: The patches of the synthesis at the previous scale $l+1$ is upsampled to obtain the current synthesis before transport U^l. A Gaussian mixture model is estimated on the set of patches P_{U^l} which constitutes the continuous distribution ν^l. On the other hand, the empirical patch distribution ν^l of the downsampled exemplar texture u^l is computed. As in the coarsest scale, ASGD is used to compute v^l and apply the optimal map T_v^l to get the transported patches P_V^l. Finally, the patches are recomposed to get the synthesis V^l at that scale.

First, U^l is obtained by upsampling the patches of the previous synthesis P_V^{l+1}. It is then used to build the continuous measure for the semi-discrete optimal transport. Furthermore, instead of the ADSN model, a Gaussian mixture model (GMM) with n_{GMM} components is estimated with the expectation-maximization algorithm to fit the patch distribution of U^l. This GMM estimation constitutes the probability measure μ^l on $\mathbb{R}^{dw^2}$ at this scale. As in the mono-scale version, ASGD is used to compute the optimal transport map T_v^l from μ^l to ν^l. Applying T_v^l on the patches of U^l (noted P_{U^l}) allows to get the transformed patches P_{V^l}. The same patch recomposition operator R can then be applied to form the synthesis V^l. This global strategy is applied for every scale except the coarsest scale $l = L - 1$.

3.3 Multiscale synthesis framework

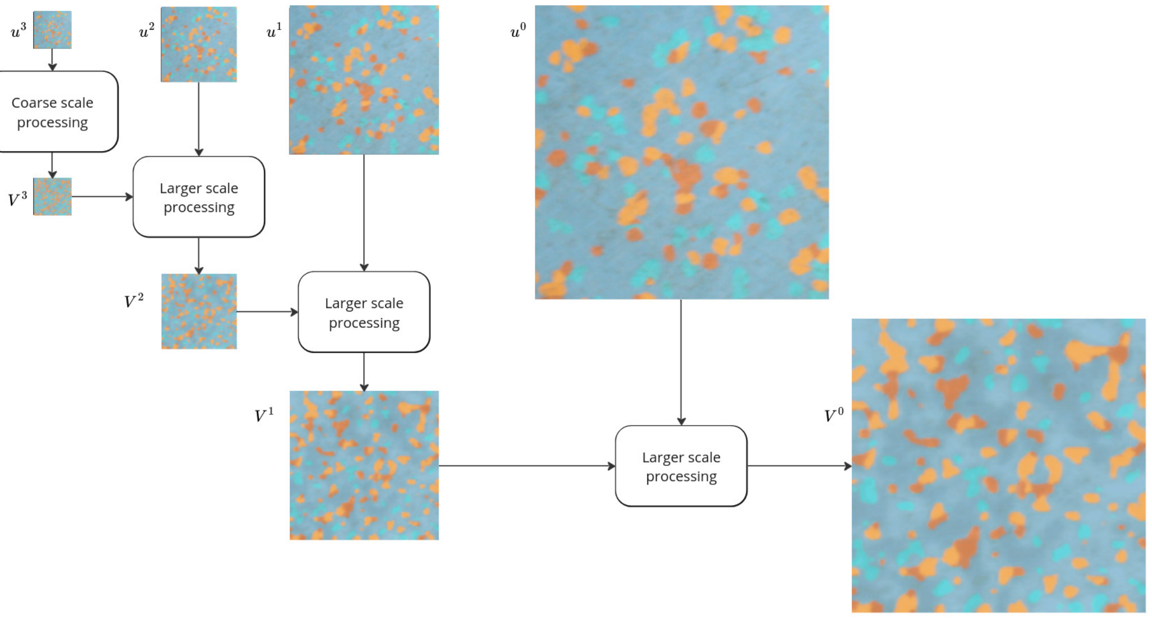

Figure 4: **Multiscale framework**: Synthesis on $L = 4$ scales from an exemplar texture. At the first scale, the coarse synthesis method described in Figure 2 is used. On all the other scales, the method described in Figure 3 is employed.

The global multiscale framework is illustrated in Figure 4, and allows to synthesize structured textures efficiently. Starting from a well-chosen Gaussian random field, successive transformations based on semi-discrete optimal transport are applied, which allows reimposing the patch distribution at different scales, thus keeping statistical guarantees. It is important to note that this round of synthesis constitutes the model estimation step. After the estimation for all scales l, the OT maps T_v^l are stored to be used for the synthesis of possibly very large images.

4 Extensions

In this section, we will propose some extensions or modifications to the actual method. The goal is to understand what factors lead to a successful synthesis and to see if computational time can be reduced. Some extensions simply consist of modifying a specific part of the original algorithm (e.g. the way we sample patches). Other extensions aim at modifying the structure of the algorithm itself with a different transportation method, such as affine transport described in this section. We will first describe those extensions, and discuss the obtained results in the next section with numerical and graphical results.

4.1 Sampling random patches instead of Gaussian modeling

For patch sampling, the original method consists of creating another Gaussian model :

- At the first scale: estimating a Gaussian model out of the ADSN sampled patches which serves as a sampling function for ASGD. Let $\mu(U_{|w})$ and $\Sigma(U_{|w})$ be the mean and covariance and X a sample, then $X \sim \mathcal{N}(\mu(U_{|w}), \Sigma(U_{|w}))$.

- At the other scales: fitting a Gaussian Mixture Model (GMM) μ^l with n_{GMM} components (4 in practice) adapted to the patches from the previous synthesis U^l, and sample from it.

What we propose is to directly sample a random patch from the current synthesis U^l. Noting P_{U^l} the finite set of patches of the current synthesis (before transport), then $X \sim \mathcal{U}(P_{U^l})$, so X follows a discrete uniform distribution. If for example the current synthesis before transport has dimension 128×128, then there are $(128-2) \times (128-2) = 15876$ possible patches to sample from. It remains in accordance with the principle of not copying patches from the original texture to have an original synthesis. Furthermore, by construction, the synthesis U^l keeps the statistics of the previous synthesis V^{l+1}.

In this adaptation, we are facing a discrete optimal transport problem as the source distribution μ is now a discrete uniform distribution, and the target distribution ν remains identical to its initial definition in equation 11. As μ is not absolutely continuous anymore, we are now optimizing a slightly different version of $H(v)$ originally defined in equation 5.

Proposition 3. *For μ and ν two discrete probability distributions with finite support S' and S respectively, the solution of the discrete optimal transport problem is obtained with the concave optimization problem:*

$$\arg\max_{v \in \mathbb{R}^S} H(v) \quad \textit{where} \quad H(v) = \sum_{x \in S'} v^c(x)\mu_x + \sum_{y \in S} v(y)\nu_y \tag{14}$$

Furthermore, H is $\mathcal{C}^1$ smooth almost everywhere, and its gradient with respect to $v(y)$ is given by

$$\frac{\partial H}{\partial v(y)} = \nu_y - \sum_{x \in Pow_v(y)} \mu_x \tag{15}$$

In this discrete setting, the optimization problem changes as the mass cannot be split. This means that the power cells might not correspond exactly to the ν-measure of the points. Instead, the transportation plan would assign entire masses from points associated with ν to points associated with μ, which would result in a transportation plan that does not fully respect the marginal distribution of ν and μ. Therefore, the previous conclusion on the fact that v is a critical point of H if and only if $\nu_y = \mu(Pow_v(y))$ for all y would not hold in the discrete case since the marginals are not respected.

Proposition 4. *For μ and ν two discrete probability distributions with finite supports, v is a critical point of H if and only if the transport plan T respects the mass at each point, i.e., $T(\nu) = \mu$.*

In other words, the transport plan ensures that the total mass transported to a point in the support of μ equals the mass of the point in the support of ν. This no longer implies that the marginals of $T(\nu)$ and μ are equal since the transport plan is not required to be a bijection between the support of ν and μ, but must respect the total mass at each point. To obtain a proper solution to this discrete optimal transport problem, one would need to add a condition ensuring that the total mass transportation is respected.

However, in practice, we can still use ASGD without regularization to find an approximate solution to the discrete optimal transport problem. As the number of points in the discrete distributions is large, especially for μ when we sample a random patch from the current synthesis U^l at each iteration, it can be seen as an approximation of a continuous distribution. Hence, using an ASGD-based approach to optimize as done in the semi-discrete case can work well. Moreover, in that case, the ASGD optimization does not aim to find a solution that perfectly matches the discrete distributions, but rather a general transport plan that minimizes the cost on average, which is sufficient for practical purposes as the exact matching distributions are not crucial.

One of the goals of this approach for patch sampling is to observe potential visual differences between those sampling methods. On the one side, we could for instance imagine that some patterns of the new synthesis are too close to those of the exemplar texture compared to the gaussianized sampling functions (which we do not wish). However, we believe that the samples from U^l are already enough randomized thanks to the Gaussian Random field modeling with ADSN on the coarsest scale and that ASGD should still converge as shown in the original paper. We will further investigate this intuition in the results section.

4.2 Proposing new methods for patch recomposition

The transformations are done locally, ie patch per patch. However, the resulting global transformation has "too many" pixel values because transformed patches overlap. To recompose the synthesis, the method uses averaging. The averaging operation is defined in eq. (13) as:

$$V(x) = \sum_{p \in \mathcal{V}(x)} \frac{1}{|\mathcal{V}(x)|} p_x.$$

We define two other recomposition operations that we wish to test. The first one consists of computing the median instead of the mean. The motivation behind this approach is to avoid potential outliers that might lead to a less visually satisfying synthesis. The median is more robust to the presence of such outliers. However, we can also expect some drawbacks. Firstly, the computational cost is lower for the mean than the median and also does not require storage of the values. Furthermore, if the superposition of the transported patch is already satisfying, then the mean might be more suitable and render a smoother visualization.

We define a second recomposition that consists of assigning weights to patches, which thus results in a weighted average. The weighted operation is then defined as:

$$V(x) = \sum_{p \in \mathcal{V}(x)} p_x \cdot \mathrm{W}_p$$

For the choices of weights, we decided to base it on the transportation cost. More precisely, the more a patch has been transported, the lower its associated weight. Suppose we take as input the transportation costs $\{C_p\}$. We apply the following operations to obtain :

$$W_p = 8 \cdot \frac{C_p - \min(C_p)}{\max(C_p) - \min(C_p)} - 4$$
$$W_p = 1 - \frac{1}{1 + e^{-W_p}}$$

The goal of applying a sigmoid function to the weights is to bring more discrimination, which we ensure by distributing the weights between -4 and 4 beforehand (corresponding to values between 0.02 and 0.98). In terms of computational cost, the above operations are relatively simple and the transportation costs are already computed beforehand.

4.3 Using affine transport

In this part, we will propose a new way to transport the patches. Instead of using the biased nearest neighbor method which is based on the (heavy) computation of v through ASGD, we use an affine transformation where we adjust the mean and covariance matrix at the start and at the end to match the two distributions. The main advantage of this method is naturally the computational cost, as we only have to compute square roots and the inverse of both covariance matrices once, as shown in eq. (16).

Definition 5. *An affine transport transformation T_{aff} between a source X and a target Y can be written as*

$$T_{aff}(x) = Ax + b \qquad (16)$$

In order to create a coherent transport plan from one distribution to another, we consider the following A and b :

$$A = \Sigma(Y)^{\frac{1}{2}} \Sigma(X)^{-\frac{1}{2}}$$
$$b = \mu(Y) - A\mu(X)$$

Here, $\Sigma(X)$ is the covariance matrix, and $\mu(X)$ is the mean of X.

Proposition 5. *With the parameters given above and the definition of T_{aff}, we guarantee the following for X of mean and covariance matrix $(\mu(X), \Sigma(X))$:*

$$\begin{cases} \mu(T_{aff}(X)) = \mu(Y) \\ \Sigma(T_{aff}(X)) = \Sigma(Y) \end{cases} \tag{17}$$

Proof. Starting from the given formulas for A and b, we aim to show that

$$\mu(T_{\text{aff}}(X)) = \mu(Y) \quad \text{and} \quad \Sigma(T_{\text{aff}}(X)) = \Sigma(Y)$$

Now, $\mu(T_{\text{aff}}(X)) = A\mu(X) + b = A\mu(X) + \mu(Y) - A\mu(X) = \mu(Y)$. For the covariance matrix, we use two results:

1. If X has covariance $\Sigma(X)$ and $A \perp\!\!\!\perp X$, then AX has covariance $A\Sigma(X)A^T$.
2. $S_n^{++}(\mathbb{R})$ is stable under inversion and taking the square root (unique in this regard).

Thus:

$$\begin{aligned} \Sigma(T_{\text{aff}}(X)) &= \Sigma(AX) \quad \text{because } b \text{ is constant} \\ &= A\Sigma(X)A^T \\ &= \Sigma(Y)^{\frac{1}{2}}\Sigma(X)^{-\frac{1}{2}}\Sigma(X)\Sigma(X)^{-\frac{1}{2}}\Sigma(Y)^{\frac{1}{2}} \\ &= \Sigma(Y)^{\frac{1}{2}}\Sigma(Y)^{\frac{1}{2}} \\ &= \Sigma(Y) \end{aligned}$$

□

In this method, instead of finding v through ASGD, we compute the affine transformation through A and b. The parameters A and b are chosen to adjust the mean and covariance of the patch distribution of U^l to the patch distribution ν^l. Then, applying the transformation to U^l gives transformed patches, that we can then map through a nearest neighbor assignment with the patches of ν^l. Affine transport has several advantages over other methods for computing optimal transformations. Firstly, it is more computationally efficient. Secondly, it is more flexible as directly brings patches back to the desired distribution. The major drawback is the precision obtained and the fact that we assume a form of linearity. Thus it might be more sensitive to potential outliers and induce poor transport for some patches.

5 Experiments

In this section, we conduct a series of experiments to evaluate the efficiency of the methods and extensions proposed above. One of the targets of our review consists of measuring the different distances between the patches of the synthesis and the target patch distribution. We also aim to look into the effectiveness of different patch recomposition methods, the impact of varying the patch size hyperparameter, as well as a comparison of the runtimes of each method. These experiments are designed not only to quantify the performance of these methodologies but also to provide qualitative insights to better understand the strengths and limitations of different approaches. For these experiments, we were able to start from the code of Texto provided by Arthur Leclaire and added our extensions. Our code as well as the tested exemplar textures are available on GitHub: https://github.com/Clement-W/TextureOptimalTransport_MVA.

5.1 Measure of distances from the target measure

Our experiments aim to test and compare our extensions with the original methods in terms of distance to the target measure that we want to approach. More specifically we compare 4 methods:

- **Texto**: the original method proposed in the article.
- **RandomPatch**: this method is described in section 4.1, we replace the continuous measure μ with a discrete measure corresponding to a random patch selected from the current synthesis U^l.

- **AffineTransport**: This method is described in section 4.3, we replace the optimal transport problem with a simple affine transformation that adjusts the mean and covariance of the source and target distributions.
- **NNProjection**: This method corresponds to a simple nearest neighbor projection of patches without adding a bias v found by ASGD.

For the following results, the hyperparameters are fixed for all methods. We use a synthesis on $L = 4$ scales with $n_{GMM} = 4$ for Texto, and a patch size of 3×3. We compare these methods using the L2 Wasserstein distance between the samples P_{V^l} (transported patches) and Y^l (samples from the target patch distribution). Noting $\hat{P}_{V^l}$ and ν^l the associated empirical distributions of patches, this distance can be written $W_2(\hat{P}_{V^l}, \nu^l)$. For these experiments, the transport distances are computed using to the POT library [1]. As stochasticity is involved in the model, we averaged the obtained distances over 10 runs, and then over the 4 scales of synthesis on each texture. We also computed confidence intervals. Figure 5 displays these averaged Wasserstein distances for 8 selected textures. Moreover, these distances are also given in Table 1 with confidence intervals.

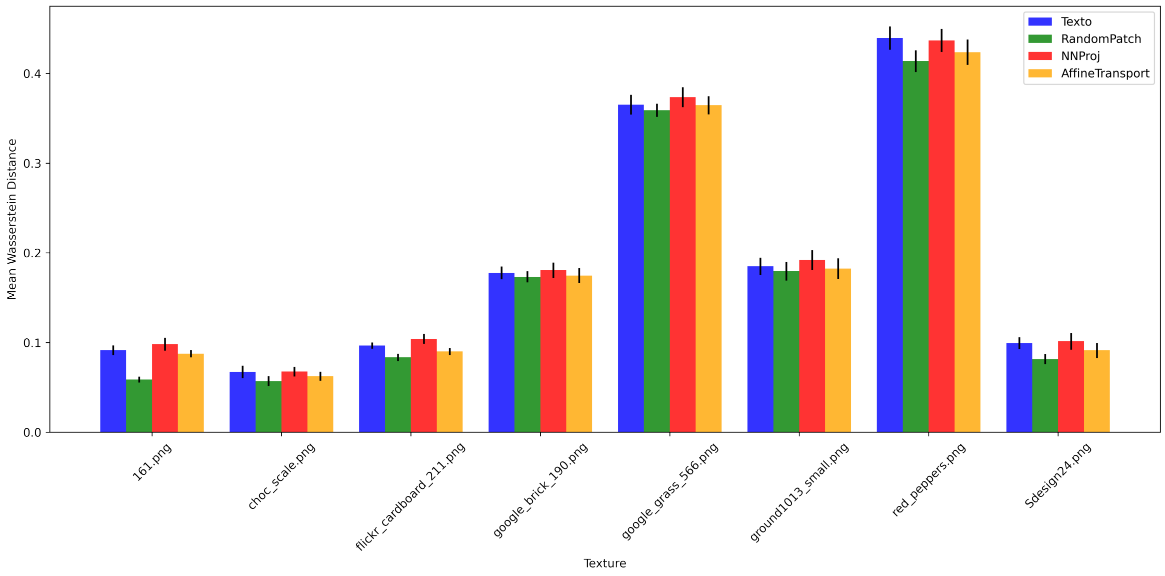

Figure 5: **Averaged Wasserstein distance** between the transported patches and the target patch distribution. The results are averaged over 10 models for each texture and method. The standard deviation is displayed with bars on the top of each histogram bar. These histograms compare the distances obtained for the 4 considered methods: Texto, RandomPatch, NNProj, and AffineTransport.

We first observe in Figure 5 that for each texture, the nearest neighbor projection provides the largest transport distance, which is expected as this method simply looks for the closest patches between the two patch distributions without paying attention to the cost of transportation. Then we observe similar results between Texto and AffineTransport. However, by looking closely at Table 1, we note that the distance from the target distribution with AffineTransport is always smaller or equal to that with Texto. Finally, for all textures, the RandomPatch method provides the lowest Wasserstein distance. We hypothesize that RandomPatch avoids approximating the patch distribution with a GMM at each scale, and thus better preserves the transported patch distributions.

To deepen our analysis and visualize the impact of these different methods on the synthesis, Figure 6 displays the synthesis obtained on a set of textures for each method. The exemplar textures are available in Appendix A. First, we observe that AffineTransport, Texto, and RandomPatch give the same visual impression for the textures ground1013_small, google_grass_566, choc_scale and 161. However, the nearest neighbor projection does not provide visually good results, except for the texture 161 which has limited variations. For the flickr_cardboard_211 texture, AffineTransort and RandomPatch both give similar results, and it seems that Texto preserves the glossiness aspect of the "painting" in a better way. This may come from the GMM modeling that captures well the lighter blurred areas, giving an

Texture name	Texto	RandomPatch	NNProj	AffineTransport
161.png	0.091 ± 0.003	**0.059 ± 0.002**	0.098 ± 0.004	0.088 ± 0.003
Sdesign24.png	0.067 ± 0.004	**0.057 ± 0.003**	0.068 ± 0.003	0.063 ± 0.003
choc_scale.png	0.097 ± 0.002	**0.083 ± 0.003**	0.104 ± 0.003	0.090 ± 0.002
flickr_cardboard_211.png	0.178 ± 0.004	**0.173 ± 0.004**	0.181 ± 0.005	0.175 ± 0.005
google_brick_190.png	0.365 ± 0.007	**0.359 ± 0.005**	0.374 ± 0.007	0.365 ± 0.006
google_grass_566.png	0.185 ± 0.006	**0.180 ± 0.006**	0.192 ± 0.007	0.182 ± 0.007
ground1013_small.png	0.439 ± 0.008	**0.414 ± 0.008**	0.437 ± 0.008	0.424 ± 0.009
red_peppers.png	0.099 ± 0.004	**0.082 ± 0.004**	0.101 ± 0.006	0.091 ± 0.005

Table 1: **Averaged Wasserstein distance** between the transported patches and the target patch distribution. The results are averaged over 10 models for each texture and method. The confidence intervals are displayed along with each mean value. The distance values are compared between the 4 considered methods: Texto, RandomPatch, NNProj, and AffineTransport. The smallest distances among these methods are emphasized in bold.

impression of glossiness. However, this remains a particularly challenging texture, as the reflections on the bottom of the exemplar texture can hardly be captured by any of these models. Finally, for the red_pepper texture, the AffineTransport seems to provide the best visual results especially because of the space between the peppers that is better preserved than with the other methods.

5.2 Study of patch recomposition operations

In this section, we aim to study the effect of patch recomposition. In the synthesis process, the patches P_{U^l} are transformed by T_v^l to make this patch distribution closer to the target distribution ν^l. Then the overlapping patches are averaged to recompose the patches into an image. An interesting question is to measure how much this recomposition R distances us from the target distribution ν^l that had been approached. To achieve this task, we can initially calculate the distance between a patch and its transported version, and then determine the distance between the transported patch and its recomposed version. This process provides insight into the extent of changes both before and after the recomposition. More rigorously, we are comparing $d_1 = \mathbb{E}\left[||P_{U^l} - P_{V^l}||_2\right]$ and $d_2 = \mathbb{E}\left[||P_{V^l} - P_{R(P_{V^l})}||_2\right]$ where $P_{R(P_{V^l})}$ corresponds to the set of patches of the recomposed images from the patches P_{V^l}. If d_2 is negligible compared to d_1, it indicates that the recomposition operation has minimal impact on the values of the transported patch. Furthermore, it suggests that even after recomposition, the patch distribution remains closely aligned with the transported patch distribution. To extend this analysis, we also compare different methods for the patch recomposition operator R. Originally, R is defined in equation 13, and corresponds to an averaging of the pixels of overlapping patches. Here, we also compare the 2 alternative methods defined in section 4.2:

- **MedianRecomp**: Compute the median instead of the mean to recompose the patches
- **WeightedAvg**: Compute a weighted average of the overlapping patches using the patch transportation costs as weights.

Figure 7 displays the distances d_1 and d_2 computed on the 4 scales and averaged over 5 models of the same texture: red pepper. For all experiments of this section, we use the method Text with $L = 3$ scales, patch size of 3×3, and 4 GMM components. We first notice that d_2 (solid lines) is always inferior to d_2 (dashed lines). However, we cannot state for sure that d_2 is negligible compared to d_1 as they have the same order of magnitude. Hence, we observe that the recomposition operator is clearly altering the pixel values, no matter the method employed between mean, median, or weighted average. Moreover, as we are comparing the pixel values, it is not easy to discriminate which of these 3 methods is more suitable to minimize the alterations of the transported patches. This first analysis helped to understand that the recomposition operation applies another non-negligible transformation to the patches. A relevant question that arises is how this recomposition is moving away the patch distribution of P_{V^l} from the target distribution ν^l. As the transport $T_v^l(x)$ transported the patches closer to ν^l, this second transformation could indeed move our patches further away from this distribution.

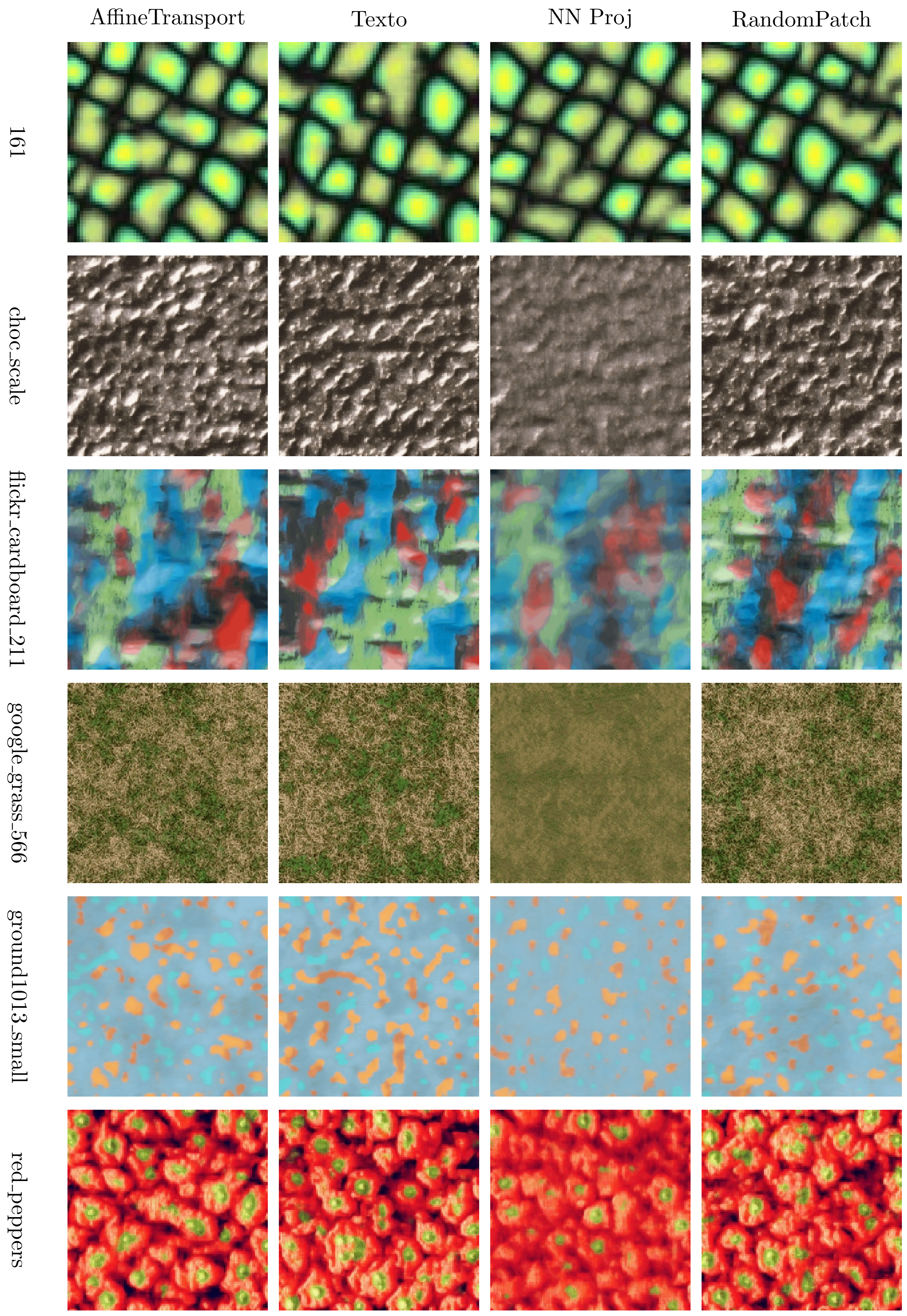

Figure 6: Synthesis of 6 textures with the four methods AffineTransport, Texto, NN Proj, and RandomPatch. All syntheses are performed with $L = 4$ scales, a patch size of 3×3, and 4 GMM components for Texto.

For this purpose, we computed the Wasserstein distance between the target distributions with the distribution of transported patches and recomposed patches:

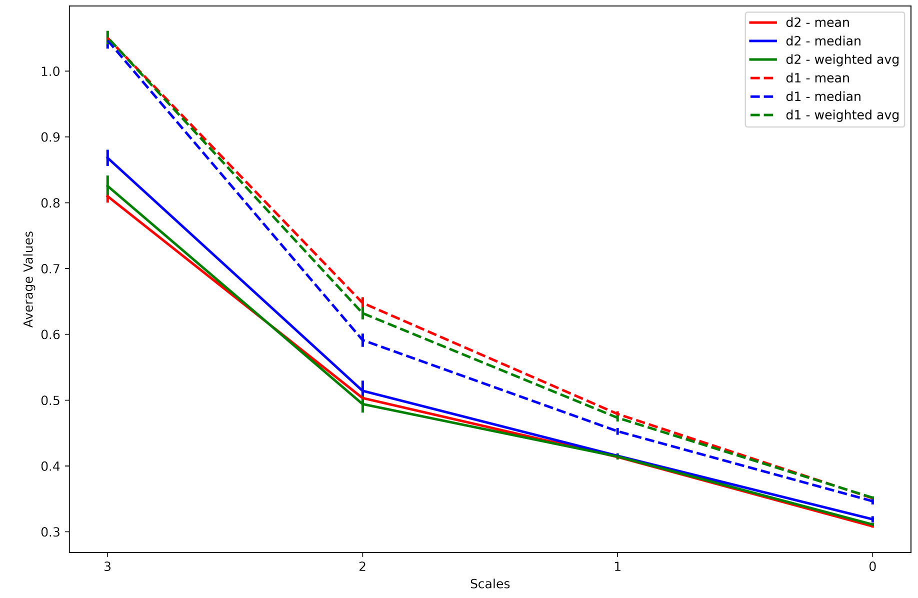

Figure 7: Comparison of distances $d_1 = \mathbb{E}\left[||P_{U^l} - P_{V^l}||_2\right]$ (solid line) and $d_2 = \mathbb{E}\left[||P_{V^l} - P_{R(P_{V^l})}||_2\right]$ (dashed lines) on 4 scales, averaged over 5 models built on the red pepper texture. The different recomposition methods R (simple average, median, weighted average) are displayed with different colors. The standard deviation is displayed with vertical lines at each scale.

- $d_{transfo} = W_2(\hat{P}_{V^l}, \nu^l)$ with $\hat{P}_{V^l}$ the associated empirical distributions to P_{V^l}
- $d_{recomp} = W_2(\hat{P}_{R(P_{V^l})}, \nu^l)$ with $\hat{P}_{R(P_{V^l})}$ the associated empirical distributions to $P_{R(P_{V^l})}$

The resulting distances are presented in figure 8. These transport distances were again averaged over

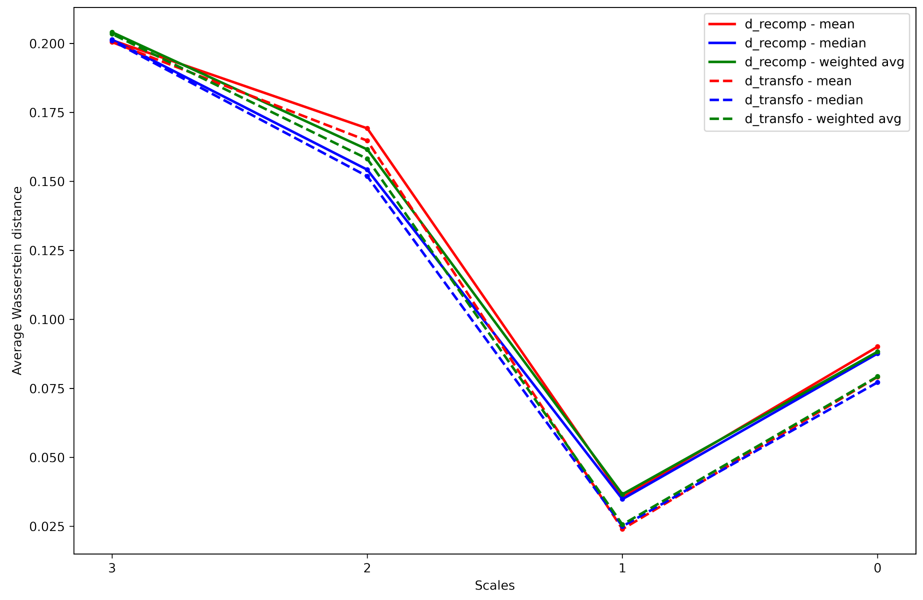

Figure 8: Comparison of Wasserstein distances $d_{transfo} = W_2(\hat{P}_{V^l}, \nu^l)$ (dashed lines) and $d_{recomp} = W_2(\hat{P}_{R(P_{V^l})}, \nu^l)$ (solid lines) on 4 scales, averaged over 5 models built on the red pepper texture. The different recomposition methods R (simple average, median, weighted average) are displayed with different colors.

5 models of the red pepper texture. First, we notice that for each recomposition method, the distance

d_{recomp} (solid lines) is always greater than $d_{transfo}$ (dashed lines). This confirms that the recomposition operation moves away the patches from the target distribution that have been approached previously by the transport. If we compare the three methods of recomposition specified with colors in Figure 8, we notice that at scale 2 there is a clear hierarchy, which seems to disappear on the next scales of synthesis. By also looking closely at the exact Wasserstein distances given in Table 2, it seems that the median recomposition helps to remain closer to the target distribution compared to the two other methods and that the simple average recomposition is the method that takes the patch distributions furthest from ν^l. To confirm these results it would be necessary to run more experiments with different textures. To better understand the impact of these recomposition methods we also tried to inspect

Method	Scale 3	Scale 2	Scale 1	Scale 0
(mean) d_transfo	0.2005 ± 0.0152	0.1647 ± 0.0111	0.0240 ± 0.0003	0.0791 ± 0.0062
(mean) d_recomp	**0.2011** ± 0.0153	0.1692 ± 0.0112	0.0355 ± 0.0013	0.0901 ± 0.0041
(median) d_transfo	0.2010 ± 0.0199	0.1518 ± 0.0053	0.0250 ± 0.0009	0.0772 ± 0.0075
(median) d_recomp	0.2014 ± 0.0200	**0.1542** ± 0.0054	**0.0348** ± 0.0015	**0.0876** ± 0.0042
(w. avg) d_transfo	0.2034 ± 0.0164	0.1582 ± 0.0128	0.0256 ± 0.0008	0.0793 ± 0.0045
(w. avg) d_recomp	0.2040 ± 0.0165	0.1616 ± 0.0135	0.0366 ± 0.0010	0.0883 ± 0.0041

Table 2: Comparison of Wasserstein distances $d_{transfo} = W_2(\hat{P}_{V^l}, \nu^l)$ and $d_{recomp} = W_2(\hat{P}_{R(P_{V^l})}, \nu^l)$ (solid lines) on 4 scales, averaged over 5 models built on the red pepper texture. The confidence interval of each averaged distance is also given. This table also compares the 3 different recomposition methods mean, median, and weighted average (w. avg). For each scale, the minimal d_recomp is emphasized in bold which shows the method that minimizes the transport distance to the target distribution.

multiple syntheses with the 3 recomposition methods. As an example, Figure 9 displays the synthesis of the red pepper texture with the 3 methods, which shows that there are no obvious differences.

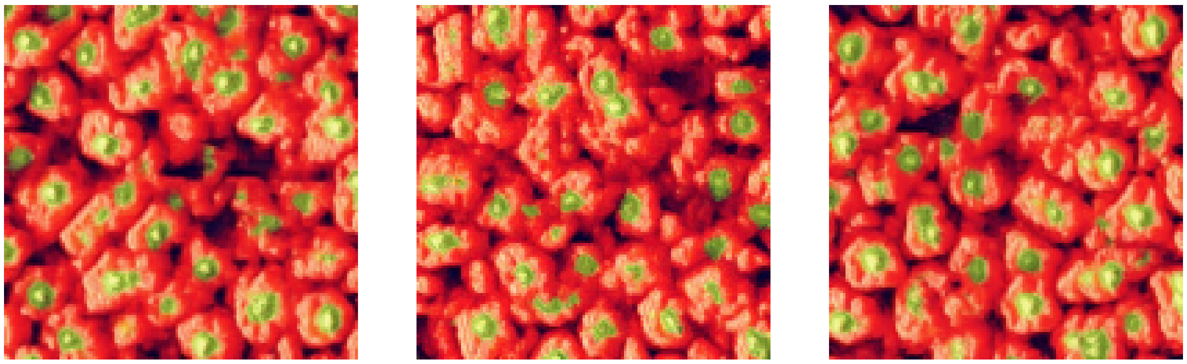
(a) Simple average (b) Median (c) Weighted Average

Figure 9: Visual Comparison of patch recomposition methods for the red pepper texture.

5.3 Increasing the patch size

Some textures contain large repetitive patterns that cannot be captured with small patch sizes such as 3×3. However, due to the computational complexity of ASGD, it is difficult to increase the patch size for Texto. For instance, increasing the patch size from 3×3 to 4×4 increases the dimension from 27 to 48. This high dimension makes ASGD longer to converge, which prevents the use of larger patch sizes in practice. However, it is interesting to compare the behavior of Texto, RandomPatch, NNProj, and AffineTransport with different patch sizes. Figure 10 displays the Sdesign24 texture synthesized by the 4 methods with 3 different patch sizes. This texture is very challenging due to the size and regularity of the repetitive pattern which is hard to capture with a small patch size. For the default patch size 3×3, we roughly observe similar results between Texto, RandomPatch, and Affine, even though Affine already captures larger patterns compared to the 2 other methods. The nearest neighbor method is hardly capturing the relevant information and let appears several holes or wrong patterns. At a patch size of 4×4, the affine transform method already produces visually close results, despite some missing parts appearing on some patterns. With a patch size of 5×5, Texto and RandomPatch produces similar results with a correct reproduction of the texture despite some missing parts. The synthesis produced with affine transport is almost perfect, we can still observe missing parts at some locations. This experiment shows that at an increased computational cost, having a larger patch size indeed improves the reconstruction quality for some textures. From our experiments, it seems that the AffineTransport method produces the closest textures to the exemplar texture in this extreme case.

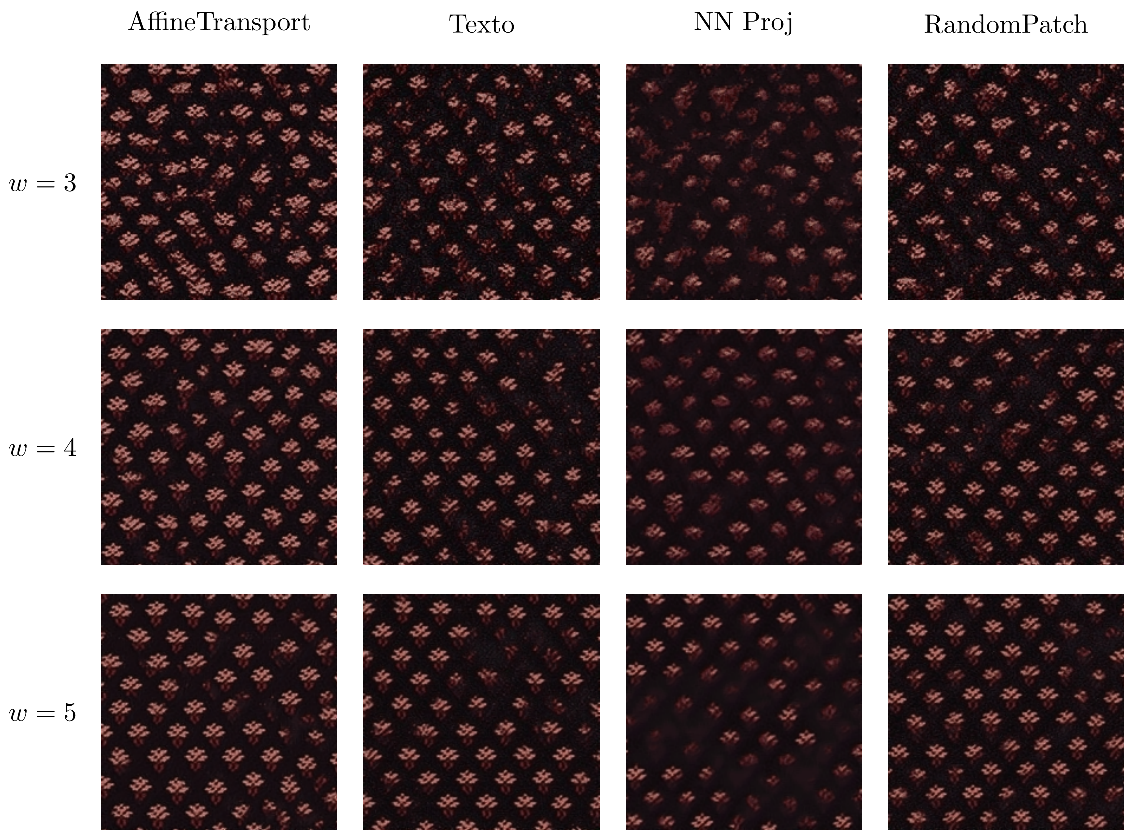

Figure 10: Synthesis of the Sdesign24 texture with the 4 methods (AffineTransport, Texto, NNProj, RandomPatch) and different patch sizes w varying from 3×3 to 5×5.

5.4 Runtime comparison

Finally, we can compare the computational time of the different methods: RandomPatch, AffineTransport, Texto, and Nearest neighbors projection. As done in section 5.1 we averaged the results on the same 10 runs. These computations were made on the same computer working with an Intel(R) Xeon(R) W-1290P CPU processor of frequency 3.70GHz. The averaged runtime in seconds with the confidence intervals is displayed in table 3. We notice that affine transport is approximately as fast as the nearest neighbor, which is expected as it is essentially equivalent, except that affine transport applies an additional step of transformation to the patches, which is very fast to estimate and compute. For Randompatch and Texto, the longer runtime is due to ASGD which aims to find the optimal v to transport μ on ν. Compared to Texto random patch is faster because instead of sampling into a GMM at each step of ASGD, a random patch is selected in the set of patches, which is very fast.

Texture (Size)	Texto	RandomPatch	NNProj	AffineTransport
161.png (64, 64)	204.24 ± 2.67	18.74 ± 0.29	1.09 ± 0.02	1.12 ± 0.01
Sdesign24.png (200, 200)	228.97 ± 1.47	39.54 ± 0.52	14.26 ± 0.27	14.45 ± 0.21
choc_scale.png (150, 150)	215.80 ± 1.45	31.42 ± 0.52	8.25 ± 0.15	8.52 ± 0.11
flickr_cardboard_211.png (300, 300)	244.52 ± 2.24	53.66 ± 0.80	27.13 ± 0.42	28.00 ± 0.54
google_brick_190.png (300, 300)	247.71 ± 2.59	54.24 ± 0.89	26.99 ± 0.38	28.25 ± 0.49
google_grass_566.png (300, 300)	250.16 ± 3.01	54.27 ± 0.78	28.25 ± 0.62	28.34 ± 0.69
ground1013_small.png (256, 256)	238.44 ± 1.45	47.70 ± 0.73	21.85 ± 0.38	21.57 ± 0.23
red_peppers.png (128, 128)	216.52 ± 1.45	27.86 ± 0.60	5.64 ± 0.10	5.89 ± 0.11

Table 3: Mean runtimes in seconds ± confidence intervals computed on 10 syntheses for each different texture and method. The calculations were made using an Intel(R) Xeon(R) W-1290P CPU processor of frequency 3.70GHz.

6 Conclusion

In this report, we began by explaining the methodology described in the paper, supplemented by illustrative examples of the algorithm in action. Based on our understanding and further discussions, we introduced extensions to the original method that offered some promising insights. The results section presents quantitative and visual evaluations to test the validity of these extensions, highlighting divergences from the original approach. In particular, we observed that the RandomPatch method remains relatively close to the original Texto method. As for the recomposition method, our examination revealed minimal alteration of results, suggesting that this step is not as core as others. In addition, the application of the affine transport strategy produced positive results, supported by both numerical measurements and visual representations. To offer a more comprehensive understanding of the impact of each method, one could include a wider range of textures, chosen for their diverse characteristics. Here, it would take too much time to compute hundreds or thousands of models for each method. In conclusion, we were very pleased to work on this topic and we hope that our efforts can bring some new insights about this paper and on patch-based texture synthesis.

References

[1] Rémi Flamary et al. "POT: Python Optimal Transport". In: *Journal of Machine Learning Research* 22.78 (2021), pp. 1–8. URL: http://jmlr.org/papers/v22/20-451.html.

[2] Bruno Galerne, Yann Gousseau, and Jean-Michel Morel. "Random phase textures: Theory and synthesis". In: *IEEE Transactions on image processing* 20.1 (2010), pp. 257–267.

[3] Bruno Galerne, Arthur Leclaire, and Lionel Moisan. "A texton for fast and flexible Gaussian texture synthesis". In: *2014 22nd European Signal Processing Conference (EUSIPCO)*. IEEE. 2014, pp. 1686–1690.

[4] Bruno Galerne, Arthur Leclaire, and Julien Rabin. "A texture synthesis model based on semi-discrete optimal transport in patch space". In: *SIAM Journal on Imaging Sciences* 11.4 (2018), pp. 2456–2493.

[5] Aude Genevay et al. "Stochastic optimization for large-scale optimal transport". In: *Advances in neural information processing systems* 29 (2016).

[6] Bela Julesz. "Textons, the elements of texture perception, and their interactions". In: *Nature* 290.5802 (1981), pp. 91–97.

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

A Textures tested

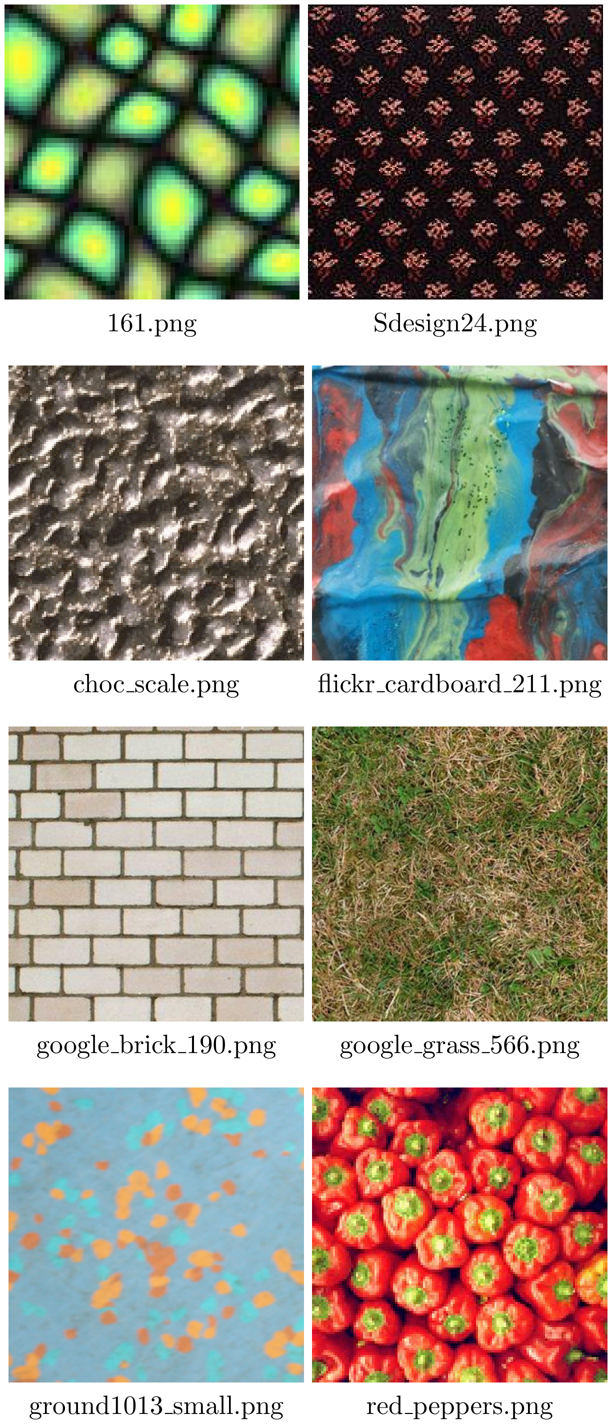

Figure 11: Exemplar textures used for our experiments.