Wasserstein Barycenter and its Application to Texture Mixing
Geometric Data Analysis report

Antoine Ratouchniak
École Normale Supérieure Paris-Saclay
antoine.ratouchniak@ens-paris-saclay.fr

1 INTRODUCTION

Synthesizing textures can have various applications in computer graphics. From a source texture, the problem involves creating a new target texture that resembles the source texture. Naturally, one of the challenges is to ensure that the synthesized texture is coherent in terms of colors, patterns, scale... This topic has been widely studied for instance with patch methods [26, 29], non-parametric methods with Markov random fields [27, 28], energy minimization [19] and more recently with deep learning models such as autoencoders, GAN [30], or diffusion models [31].

In this work, we will study thoroughly the method proposed in [1], try to reproduce the results, and attempt to enhance the robustness and efficiency of the proposed approach. We shall begin by providing essential definitions for the Wasserstein distance and its sliced version. Subsequently, we will show how this approximation can be applied and its utility in texture mixing using the steerable pyramid [15]. We will also propose new methods for texture mixing. Finally, we will discuss the limitations and conclude. The source code is publicly available at https://github.com/AntoineRtk/TextureMixing.

2 BACKGROUND

2.1 Wasserstein distance

Let Ω be an arbitrary space, $\mathcal{P}_p(\Omega)$ be the set of probability distributions with finite p-th moment and $\mu \in \mathcal{P}_p(X)$, $\nu \in \mathcal{P}_p(Y)$ two measures defined on spaces $X, Y \subset \mathbb{R}^d$. We remind $\mathcal{P}_p(\Omega) = \{\mu \in \mathcal{P}(\Omega) \mid \int_\Omega \mu(x)^p dx < \infty\}$.
The Wasserstein distance of order p is defined as:

$$\mathbf{W}_p(\mu,\nu) = \left(\inf_{\pi\in\Pi(\mu,\nu)} \int_{X\times Y} d(x,y)^p d\pi(x,y)\right)^{1/p} \tag{1}$$

where d is a cost function and Π is the set of all couplings between μ and ν, which is a joint probability whose marginals are μ and ν. If μ and ν do have a finite moment of order p, then, $\mathbf{W}_p$ is a metric [2]. For the rest of this work, we will use $d(x,y) = |x-y|$.

From now on, we will also consider discrete probability distributions α, β of same cardinal $\#X = \#Y = N$ in $\mathbb{R}^d$, represented as point clouds. Formally,

$$\mu = \frac{1}{N}\sum_{i=1}^{N} \delta_{x_i} \quad \nu = \frac{1}{N}\sum_{i=1}^{N} \delta_{y_i}$$

where δ_x is the Dirac measure in x with each coordinate being unique, $(x_i), (y_j) \in \mathbb{R}^d$. Given this setting, we will take into account the permutation of the values which is

$$X_\sigma = \{(X_{\sigma(i)})_{i=1}^N \mid \sigma \in \mathfrak{S}_N\}$$

where $\mathfrak{S}_N$ denotes the set of permutations of the N elements. Therefore, each point has one unique assignment, so we can rewrite the problem (1) as:

$$\mathbf{W}_p(\mu,\nu) = \mathbf{W}_p(X,Y) = \left(\min_{\sigma\in\mathfrak{S}_N} \sum_{i=1}^{N} d(X_i, Y_{\sigma(i)})^p\right)^{1/p} \tag{2}$$

Computing (2) directly with a brute force approach would result in a $O(N \times N!)$ complexity. Instead, one can consider the following minimization problem:

$$\begin{aligned} \min_{P\in\mathbb{R}^{N\times N}} \quad & \sum_{(i,j)\in\{1,\dots,N\}^2} P_{i,j} d(x_i,y_j)^p \\ \text{s.t.} \quad & \sum_{i=1}^{N} P_{i,j} = 1 \quad \forall j \in \{1,\dots,N\} \\ & \sum_{j=1}^{N} P_{i,j} = 1 \quad \forall i \in \{1,\dots,N\} \\ & P_{i,j}(1-P_{i,j}) = 0 \; \forall (i,j) \in \{1,\dots,N\}^2 \end{aligned} \tag{3}$$

which is in a standard form that can be solved with a linear programming algorithm with a complexity of about $O(N^{2.5}\log N)$ [4]. This minimization could also be seen as a complete bipartite graph problem and thus solved by using the Hungarian algorithm [5], but with a higher complexity.
Finally, the 1D case can be solved very efficiently [3], the optimal assignment minimizing (2) is:

$$\sigma^* = \sigma_Y \cdot \sigma_X^{-1} \tag{4}$$

where σ_X and σ_Y are respectively the permutations ordering the point clouds X_σ and Y_σ. Equivalently, we can sort both point clouds and compare them, which can be computed with a $O(N\log N)$ complexity.

2.2 Wasserstein barycenter

Wasserstein barycenters, initially introduced in [6] are constructed through an analogy with barycenters in the Euclidean space. Indeed, the Wasserstein barycenters of a collection of random variables are defined as minimizers of a weighted sum of Wasserstein distances. The Wasserstein barycenter for a sequence of point clouds $\{X_i \in \mathbb{R}^d\}_{i=1}^n$ is formally defined as:

$$\mathrm{Bar}(\lambda_i, X_i)_{1\le i\le n} = \operatorname*{argmin}_X \sum_{i=1}^{n} \lambda_i \mathbf{W}_p(X, X_i) \tag{5}$$

where $(\lambda_i)_{1\le i\le n} = \{(\lambda_i)_{1\le i\le n} \in \mathbb{R}_+ \mid \sum_{i=1}^{n} \lambda_i = 1\}$. This can intuitively be seen as a point cloud that is close to all the other point clouds in the Wasserstein sense.

2.3 Sliced-Wasserstein distance

As seen at the beginning of this section, computing the Wasserstein distance can be computationally expensive as the number of samples N grows. For this reason, there is another approach, the Sliced-Wasserstein distance [8] that leverages two ideas: projecting a distribution in one dimension by the Radon transform [7] and using the closed-form expression for these two distributions to solve the minimization problem. The Radon transform aims to reduce a multi-dimensional distribution to a one-dimensional representation through linear projections. Let $\mathbb{S}^{d-1} = \{\theta \in \mathbb{R}^d \mid \|\theta\|_2 = 1\}$ be the unit sphere in $\mathbb{R}^d$, where θ is uniformly distributed and $I \in L^1(\mathbb{R}^d)$, we define the Radon transform as:

$$\mathcal{R}I(t,\theta) = \int_{\mathbb{R}^d} I(x)\delta(t - \langle x, \theta\rangle)dx \quad \forall (t,\theta) \in \mathbb{R} \times \mathbb{S}^{d-1} \tag{6}$$

Note that $\mathcal{R}(\cdot,\theta)$ is continuous for a fixed θ. Since we are dealing with point clouds, the Sliced-Wasserstein distance of order p is:

$$\mathbf{SW}_p(X,Y) = \int_{\theta\in\mathbb{S}^{d-1}} \mathbf{W}_p(\langle X,\theta\rangle, \langle Y,\theta\rangle)d\theta \tag{7}$$

where $\langle X,\theta\rangle$ is obtained by $\mathcal{R}\alpha(\cdot,\theta)$ in a discrete setting. By doing this, it brings us back to the optimal assignment problem in 1 dimension (4). The Sliced-Wasserstein distance also benefits from good properties such as being a distance [9, 10]. In practice, computing (7) is intractable, instead, we use a Monte-Carlo scheme with K random projections on the unit sphere $\mathbb{S}^{d-1}$ such that:

$$\widehat{\mathbf{SW}}_p = \frac{1}{L}\sum_{i=1}^{L} \mathbf{W}_p(\langle X,\theta\rangle, \langle Y,\theta\rangle)$$

where $L \geq 1$ is the number of projections. It has been noticed that the number of projections L should be comparable to the number of features d [13, 14]. However, in practice, we found that using $L = 1$ was sufficient. Other alternatives exist such as the be used such as the Max-Sliced Wasserstein [11].

Analogically, we define the Sliced-Wasserstein barycenter by:

$$\widetilde{Bar}(\lambda_i, X_i)_{1\leq i\leq n} = \operatorname*{argmin}_X \sum_{i=1}^{n} \lambda_i \mathbf{SW}_p(X, X_i) \tag{8}$$

2.4 Applications to point clouds

Remaining in the same parametrization with two point clouds X and Y, we seek to minimize (7).

Stochastic gradient descent: by smoothness of the Sliced-Wasserstein [3], one can use the stochastic gradient descent with Newton step. It yields to compute:

$$X^{(t+1)} = X^{(t)} - \eta \nabla^2_{X^{(t)}} \mathbf{SW}_2^2(X^{(t)}, Y) \nabla_{X^{(t)}} \mathbf{SW}_2^2(X^{(t)}, Y) \tag{9}$$

where

$$\begin{aligned}
\nabla_{X^{(t)}} \mathbf{SW}_2^2(X^{(t)}, Y) &= \nabla_{X^{(t)}} \int_{\theta\in\mathbb{S}^{d-1}} \mathbf{W}_2^2(\langle X^{(t)},\theta\rangle, \langle Y,\theta\rangle)d\theta \\
&= \int_{\theta\in\mathbb{S}^{d-1}} \nabla_{X^{(t)}} \min_{\sigma\in\mathfrak{S}_N} \|\langle X^{(t)},\theta\rangle - \langle Y,\theta\rangle_\sigma\|_2^2 d\theta \\
&= \int_{\theta\in\mathbb{S}^{d-1}} \nabla_{X^{(t)}} \|\langle X^{(t)} - Y,\theta\rangle_{\sigma^*}\|_2^2 d\theta \\
&= \int_{\theta\in\mathbb{S}^{d-1}} 2\langle X^{(t)} - Y,\theta\rangle_{\sigma^*}\theta^T d\theta
\end{aligned}$$

where σ^* is the solution defined in (4).

and

$$\nabla^2_{X^{(t)}} \mathbf{SW}_2^2(X^{(t)}, Y) = 2\theta\theta^T$$

We choose $\eta = 1$. We denote $\mathcal{P}_{\mathbf{SW}_2^2}(X,Y)$ the minimization after gradient descent.

Similarly, $\nabla_{X^{(t)}} \widetilde{Bar}(\lambda, X_i)_{1\leq i\leq n} = \sum_{i=1}^{n} \lambda_i \nabla_{X^{(t)}} \mathbf{SW}_2(X^{(t)}, X_i)$ and we suppose that $\widetilde{Bar}(\lambda, X_i)_{1\leq i\leq n}$ is obtained by gradient descent for the rest of this work. There have recently been theoretical guarantees of convergence of $\widetilde{Bar} \overset{t\to\infty}{\longrightarrow}$ Bar [12]. Examples can be seen in figures 1 and 5.

3 TEXTURE MIXING

3.1 Steerable pyramid

The steerable pyramid introduced by Heeger and Bergen [15], enables the synthesis of textures with high fidelity. This algorithm operates as follows: starting from a source image, a high-pass and low-pass filter are applied to the image. The low-pass image is subsequently subsampled by a factor of 2 and decomposed into multiple orientations using bandpass-oriented filters [16, 17]. We can mention that the representations benefit shift invariance from low-pass filtering. These operations can be repeated in a cascade a certain number of times, allowing the capture of meaningful structural information. By applying various adjustments at each scale and orientation, the algorithm can reconstruct and generate a final result that closely resembles a target image. Formally, let S be the number of scales and Q the number of orientations, given an input image, the pyramid will produce $S \times Q + 2$ images, including:

(1) a high-frequency image at the first scale
(2) $S \times Q$ high-pass oriented images at different scales
(3) a low-frequency image at the final scale

We describe the use of steerable filters introduced in [16]. From now on, let $u \in \mathbb{R}^{N\times N}$ be a discrete image of size $N \times N$, N being even. The discrete Fourier transform of u is:

$$\mathcal{F}_{(k,l)}\{u\} = \sum_{p=0}^{N-1}\sum_{q=0}^{N-1} u_{p,q} e^{-2i\pi kp/N} e^{-2i\pi lq/N} \quad \forall (k,l) \in \{\frac{-N}{2}, \dots, \frac{N}{2}-1\}^2 \tag{10}$$

and the inverse discrete Fourier transform of $v \in \mathbb{C}^{N\times N}$:

$$\mathcal{F}^{-1}_{(k,l)}\{v\} = \frac{1}{N^2}\sum_{p=\frac{-N}{2}}^{\frac{N}{2}-1}\sum_{p=\frac{-N}{2}}^{\frac{N}{2}-1} v_{p,q} e^{2i\pi kp/N} e^{2i\pi lq/N} \quad \forall (k,l) \in \{0,\dots,N-1\}^2 \tag{11}$$

We also need to define the operator to transform cartesian coordinates to polar coordinates as:

$$\begin{aligned}
\rho : [-\pi,\pi]^2 &\longrightarrow \mathbb{R}\times]-\pi,\pi] \\
(x,y) &\longmapsto (r,\theta) = \begin{cases} (|x|,\pi) & \text{if } y = 0 \text{ and } x \leq 0 \\ \left(\sqrt{x^2+y^2}, 2\arctan\left(\frac{y}{x+\sqrt{x^2+y^2}}\right)\right) & \text{otherwise} \end{cases}
\end{aligned}$$

The low-pass, high-pass and oriented filters we use are polar-separable in the Fourier domain:

$$L(r,\theta) = L(r) = \begin{cases} 1 & \text{if } r \leq \frac{\pi}{4} \\ \cos(\frac{\pi}{2}\log_2(\frac{4r}{\pi})) & \text{if } \frac{\pi}{4} \leq r \leq \frac{\pi}{2} \\ 0 & \text{if } r \geq \frac{\pi}{2} \end{cases}$$

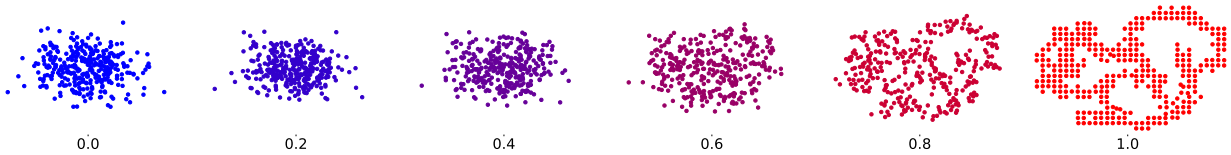


Figure 1: Sliced-Wasserstein distance from a Gaussian point cloud to a squirrel point cloud

$$H(r,\theta) = H(r) = \begin{cases} 0 & \text{if } r \leq \frac{\pi}{4} \\ \cos(\frac{\pi}{2}\log_2(\frac{2r}{\pi})) & \text{if } \frac{\pi}{4} \leq r \leq \frac{\pi}{2} \\ 1 & \text{if } r \geq \frac{\pi}{2} \end{cases}$$

$$G_k(\theta) = \begin{cases} 2^{k-1}\frac{(K-1)!}{\sqrt{K(2(K-1))!}}\left[\cos(\theta - \frac{k\pi}{K})\right]^{K-1} & \text{if } |\theta - \frac{k\pi}{K}| < \frac{\pi}{2} \\ 0 & \text{otherwise} \end{cases}$$

Finally, the lowpass residual bands and oriented subbands are defined as:

$$H_0(r) = H\left(\frac{r}{2}\right)$$
$$L_0(r) = L\left(\frac{r}{2}\right)$$
$$B_k(r,\theta) = H(r)G_k(\theta)$$

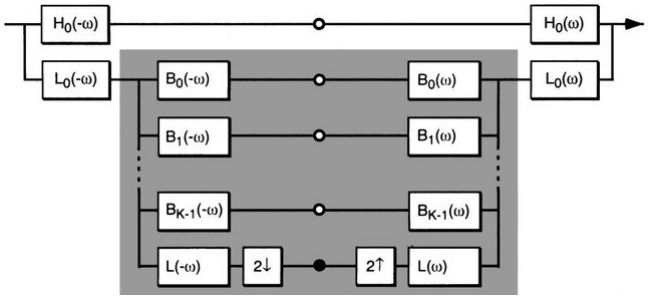


Figure 2: Diagram for the steerable pyramid. Illustration from [15]. The input image is decomposed into low and high frequencies, and the low-pass subbands are divided into multiple oriented bandpass subbands. The image is finally reconstructed by upsampling and multiplying by its according filter.

The key idea to prove the previous propositions is the following: the Fourier transform of the n-th derivative is $\mathcal{F}_k\{\frac{\partial^n}{\partial x^n}f(x)\} = (-ik)^n\mathcal{F}_k\{f(x)\}$ (hence the power K). One can also rely on the fact that the Fourier transform of the rotation is the rotation of the Fourier transform $\mathcal{F}_k\{f \cdot R(x)\} = \mathcal{F}_{R\cdot k}\{f(x)\}$ where $f \in L^1(\mathbb{R})$ is an arbitrary function and R is a rotation matrix. We refer to [18] for further explanations. We display an example in figure 3.

3.2 Texture synthesis

The steerable pyramid defined in the previous subsection allows us to perform texture synthesis. We first describe the algorithm used for texture synthesis in Algorithm 1.

The function *BuildPyr* constructs the steerable pyramid shown on the left-hand side of figure 2. The function *ReconstructPyr* is employed to reconstruct the texture. It is worth mentioning that

Algorithm 1: Texture mixing

Input: $\{T_i\}_{i=1}^n$ texture exemplars, $\{\lambda_i\}_{i=1}^n$ weights, S number of scales, Q number of orientations, N_{iter} number of iterations
Output: A synthesized texture
Data: Texture $\sim \mathcal{N}(0, I)$ // texture follows a Gaussian noise

1 **for** $i = 1, \dots, n$ **do**
2 Pyramids[i] = BuildPyr(T_i) // pyramids for ex. textures
3 Barycenter = $\widetilde{Bar}(\lambda_i, T_i)_{1\leq i\leq n}$ // barycenter of ex. textures
4 **for** $i = 1, \dots, S \times Q + 2$ **do**
5 BarPyramids[i] = $\widetilde{Bar}(\lambda_i, \text{Pyramids[i]})_{1\leq i\leq n}$
6 **for** $i = 1, \dots, N_{iter}$ **do**
7 TexturePyramid = BuildPyr(Texture)
8 **for** $j = 1, \dots, S \times Q + 2$ **do**
9 TexturePyramid[j] = $\mathcal{P}_{\text{SW}_2^2}$(TexturePyramid[j], BarPyramids[j])
10 Texture = ReconstructPyr(TexturePyramid)
11 Texture = $\mathcal{P}_{\text{SW}_2^2}$(Texture, Barycenter) // color transfer

all the projections are done on the real part of the IDFT, defined in (11). Proceeding with the Sliced-Wasserstein distance, we also do not need the PCA that was proposed in the original article to do the color transfer. In practice, due to the abundance of pixels, line 7 of Algorithm 1 can simply be the optimal assignment (4) with the target barycenter. We will use by default $S = 4$ and $Q = 4$. We first show some basic examples in figures and 6a and 6b. It is important to state that during all the experiments, the quantity $\left[\nabla_{X^{(t)}}\text{SW}_2^2(X^{(t)}, Y))\right]^T\left[\nabla^2_{X^{(t)}}\text{SW}_2^2(X^{(t)}, Y)\right]^{-1}\left[\nabla_{X^{(t)}}\text{SW}_2^2(X^{(t)}, Y)\right]$ decreases but never converges.

For now, the algorithm can synthesize texture at a high level. However, capturing details can be difficult due to the high number of samples since we do not take into account the correlation between the pixels. The original article proposed using patches to enforce coherence in close neighborhoods.

Patches: as proposed in the article, one can use patches to impose a correlation between the different parts of the images. However, it is not mentioned in the article what strategy was used to do so. Therefore, we can think of two possibilities: overlapping and non-overlapping.

Overlapping patches: let τ be the patch size, O the overlap, $\omega = \{1, \dots, \frac{N-\tau+O}{\tau-O}\}^2$ the patch domain and $p_k(u)$ the k^{th} patch of an image u where $k \in \omega$. At each scale and orientation j, we first define

BarPyramidPatch$[j] = \widetilde{Bar}(\lambda_i, (p_w(u_i))_{1\leq w\leq M})_{1\leq i\leq n}$ where u_i is the i^{th} image and M is the number of patches.
Then, for each iteration, similarly to Algorithm 1, we let $p_k(\text{Texture}[j]) = \mathcal{P}_{\mathbf{SW}_2^2}(p_k(\text{TexturePyramid}[j]), \text{BarPyramidPatch}[j])$ for every patch and all scales and orientations. Since the patches are overlapping, we reconstruct the images by averaging the overlapping parts with:

$$\text{Reconstructed}(x) = \sum_{k\in\omega} \frac{1}{|\omega|}\mathbf{1}_\omega(k)p_k(u) \quad x \in \{1,\dots,N\}^2$$

Examples can be seen in figures (7b) and (8b) with different patch sizes.
Non-overlapping patches: the process is the same, except that the patch domain is now $\omega = \{1,\dots,\frac{N}{\tau}\}^2$ and the reconstruction is done patch-wise. Examples can be seen in figures (7a) and (8a) with different patch sizes. Visually, a higher patch size helps to capture details. Nevertheless, it is not obvious whether overlapping helps or not. We also experiment with varying the number of scales and orientations in figure 9 with a patch size of 8 and no overlap. Increasing the number of scales/orientations did not seem to significantly impact the results. Finally, our results slightly differ from the original article.

Spectrum constraint: following the work of Tartavel et al. [19], one may be interested in matching the Fourier spectrum of two textures. Indeed, we recall that $\forall (k,l) \in \{\frac{-N}{2},\dots,\frac{N}{2}-1\}$, $|\mathcal{F}_{(k,l)}\{u\}|^2 = \mathcal{F}_{(k,l)}\{u * u\}$ by the convolution theorem where $\mathcal{F}\{u\}$ is the DFT, introduced earlier (10). Thus, we see that $|\mathcal{F}\{u\}|^2$ encodes the correlation of an image. For this reason, they decided that for two images u, $u_{\text{ref}} \in \mathbb{R}^{N\times N}$ where u_{ref} is the reference image, u could have the same phase φ. This reads:

$$\exists \varphi \in \mathbb{R}^2 \text{ such that } \mathcal{F}_{(k,l)}\{u\} = e^{i\varphi^T\binom{k}{l}}\mathcal{F}_{(k,l)}\{u_{\text{ref}}\}$$
$$\forall (k,l) \in \{\tfrac{-N}{2},\dots,\tfrac{N}{2}-1\}^2$$

It has been proved that we can minimize $\|u_{(k,l)} - u_{(k,l)_{\text{ref}}}\|_2^2$ and keeping the same amplitude for u by setting:

$$\mathcal{F}_{(k,l)}\{u_{\text{new}}\} = \frac{\mathcal{F}_{(k,l)}\{u\}\cdot\mathcal{F}_{(k,l)}\{u_{\text{ref}}\}}{|\mathcal{F}_{(k,l)}\{u\}\cdot\mathcal{F}_{(k,l)}\{u_{\text{ref}}\}|} \quad \forall (k,l) \in \{\frac{-N}{2},\dots,\frac{N}{2}-1\}^2 \tag{12}$$

where $\cdot$ denotes the hermitian product. We decided to match the spectrums when initializing the texture. We display some examples in figure 10. It is interesting to notice that the structures of the synthesis are better organized, especially for the checkboard. Nevertheless, it remains difficult to capture details. We can draw a parallel between the spectrum constraint and the work of Portilla et al. [20] with the auto-correlation, that was suggested at the end of the article [1]. In this paper, they maximized the local auto-correlation of an image to better describe the structure and periodicity of natural textures.

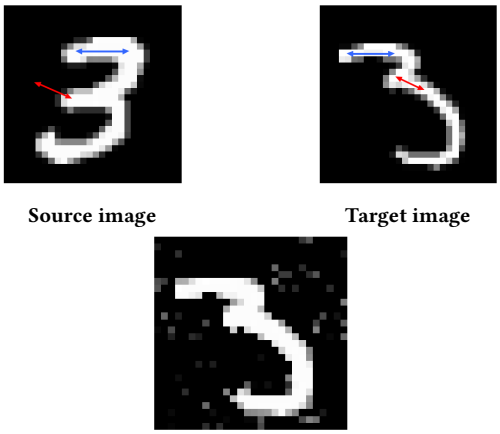
Gromov-Wasserstein for cross-correlation: the cross-correlation adjustment of the magnitudes [20] was also a characteristic of the synthesis. The idea is to impose a correlation at different scales of the pyramids, between the orientations. Instead, we propose to use the Gromov-Wasserstein distance [21] that allows us to compare two distributions not necessarily relying on the same metric space. In our case, we will consider patches of same size at the same scale but different orientations. The intuition is that for small patches, we want to minimize the distortion between the orientations. We show an example in figure 4. We use the same notations as the first section. Let Σ_1, Σ_2 be two geometric domains and $C_X : \Sigma_1\times\Sigma_1 \to \mathbb{R}_+, C_Y : \Sigma_2\times\Sigma_2 \to \mathbb{R}_+$ pairwise distances between two points in spaces X and Y respectively. The Gromov-Wasserstein distance of order p is defined as:
$\mathbf{GW}_p((\mu, C_X),(\nu, C_Y)) =$

$$\left(\inf_{\pi\in\Pi(\mu,\nu)} \iint_{\Sigma_1,\Sigma_2} (C_X(x,y) - C_Y(x',y'))^p d\pi(x,y)d\pi(x',y')\right)^{1/p} \tag{13}$$

Since we are working with point clouds, this recasts to:
$\mathbf{GW}_p((\mu, C_X),(\nu, C_Y)) = \mathbf{GW}_p(C_X, C_Y) =$

$$\left(\min_{\sigma\in\mathfrak{S}_N} \sum_{(i,j)\in\{1,\dots,N^2\}} (C_X(x_i,y_j) - C_Y(x_{\sigma(i)},y_{\sigma(j)}))^p\right)^{1/p} \tag{14}$$

where X and Y are patches. It sounds reasonable to say that for a relatively small patch size, for instance, 8, there should not be too much distortion between the orientations. We show the intuition in figure 4. Let $S = Q = 4$ orientations and scales, the process we are doing is the following: we choose a random orientation among the 4 at each scale, and impose that for the 3 other ones, they must have an optimal permutation minimizing the Gromov-Wasserstein with the chosen one. We use POT [22] to minimize this distance using the conditional gradient algorithm. After several attempts, we also found that this optimal permutation should only be applied at the coarsest scale to be beneficial. We display examples in figure 11. We chose to use the L^2 distance as a pairwise distance, as we obtained similar results with a Gaussian kernel. These experiments could be discussed and improved.



Source image **Target image**

Source image after transportation by Gromov-Wasserstein

Figure 4: Gromov-Wasserstein transportation from a source to target image (MNIST [23]). Blue arrows indicate a low distance and red arrows a high one.

4 DISCUSSIONS AND LIMITATIONS

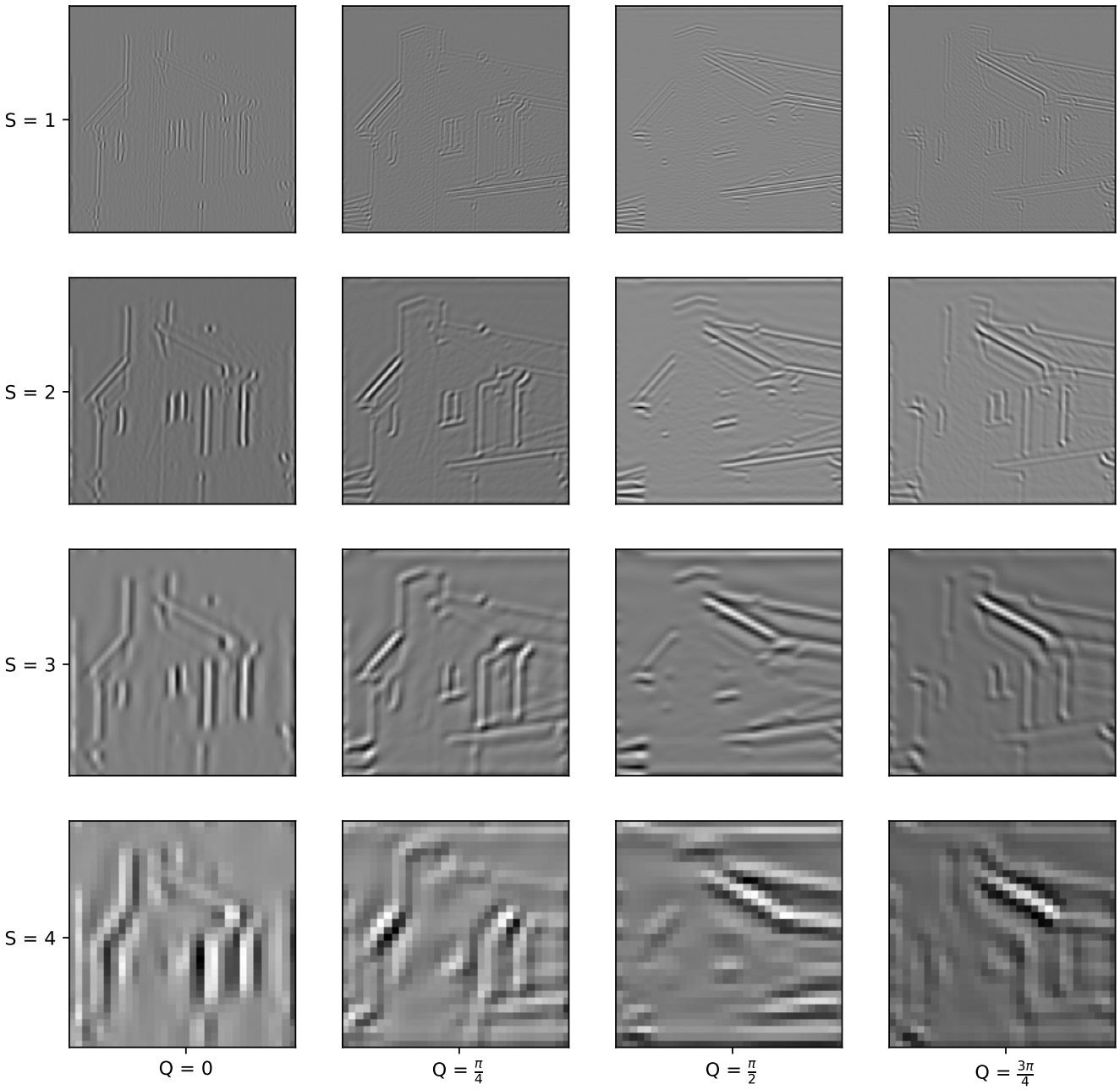
We did not address certain issues, such as inpainting, which involves generating or reconstructing missing pixels in an image, potentially within a contextual framework. It is important to note that some aspects were not very clear or missing, as they were not explicitly mentioned in the article. For example, details like the number of projections, the iterations used for texture mixing, the size of the images, and the strategy employed for patches were not communicated. Finally, the article did not provide the code either, which is a concern in today's research, as it hinders reproducibility.

5 CONCLUSION

The Wasserstein distance has garnered significant attention from researchers and is at the forefront of numerous studies today. For example, one can mention the Wasserstein GAN that can produce really interesting generations [32]. As evoked in the introduction, more recently, image generation can be done with neural networks. In 2020, Vacher et al. [25] introduced a deep convolutional model leveraging natural geodesics to perform texture mixing. To conclude, we showed with our results that the Wasserstein distance was a powerful tool even though it suffers from high dimensionality.

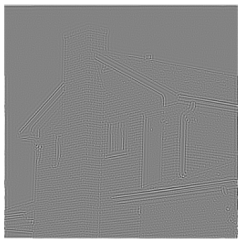
REMERCIEMENTS

Je remercie Jonathan Vacher pour l'aide qu'il m'a apporté pour transcrire le code de la pyramide de Heeger-Bergen disponible sur IPOL en Python afin de mener à bien ce travail.

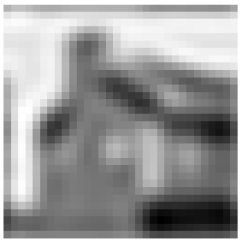


(a) Low-pass oriented subbands

Original image



High-pass residual at first scale



Low-pass residual at last scale

Figure 3: Heeger-Bergen pyramid decomposition of an image. S indicates the scale and Q the orientation.

REFERENCES

[1] Rabin, J., Peyré, G., Delon, J., & Bernot, M. (2012). Wasserstein barycenter and its application to texture mixing. In Scale Space and Variational Methods in Computer Vision: Third International Conference, SSVM 2011, Ein-Gedi, Israel, May 29–June 2, 2011, Revised Selected Papers 3 (pp. 435-446). Springer Berlin Heidelberg.

[2] Villani, C. (2009). Optimal transport: old and new (Vol. 338, p. 23). Berlin: springer.

[3] Peyré, G., & Cuturi, M. (2017). Computational optimal transport. Center for Research in Economics and Statistics Working Papers, (2017-86).

[4] Burkard, R., Dell'Amico, M., & Martello, S. (2012). Assignment problems: revised reprint. Society for Industrial and Applied Mathematics.

[5] Kuhn, H. W. (1955). The Hungarian method for the assignment problem. Naval research logistics quarterly, 2(1-2), 83-97.

[6] M. Agueh and G. Carlier. Barycenters in the wasserstein space. SIAM Journal on Mathematical Analysis, 43(2):904–924, 2011.

[7] Helgason, S. (2011). Integral geometry and Radon transforms (p. 3). New York: Springer.

[8] Bonneel, N., Rabin, J., Peyré, G., & Pfister, H. (2015). Sliced and radon wasserstein barycenters of measures. Journal of Mathematical Imaging and Vision, 51, 22-45.

[9] Bonnotte, N. (2013). Unidimensional and evolution methods for optimal transportation (Doctoral dissertation, Université Paris Sud-Paris XI; Scuola normale superiore (Pise, Italie)).

[10] Kolouri, S., Zou, Y., & Rohde, G. K. (2016). Sliced Wasserstein kernels for probability distributions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 5258-5267).

[11] Deshpande, I., Hu, Y. T., Sun, R., Pyrros, A., Siddiqui, N., Koyejo, S., ... Schwing, A. G. (2019). Max-sliced wasserstein distance and its use for gans. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 10648-10656).

[12] Tanguy, E., Flamary, R., & Delon, J. (2023). Properties of discrete sliced Wasserstein losses. arXiv preprint arXiv:2307.10352.

[13] Dai, B., & Seljak, U. (2020). Sliced iterative normalizing flows. arXiv preprint arXiv:2007.00674.

[14] Nguyen, K., Ho, N., Pham, T., & Bui, H. (2020). Distributional sliced-Wasserstein and applications to generative modeling. arXiv preprint arXiv:2002.07367.

[15] Heeger, D. J., & Bergen, J. R. (1995, September). Pyramid-based texture analysis/synthesis. In Proceedings of the 22nd annual conference on Computer graphics and interactive techniques (pp. 229-238).

[16] Freeman, W. T., & Adelson, E. H. (1991). The design and use of steerable filters. IEEE Transactions on Pattern analysis and machine intelligence, 13(9), 891-906.

[17] Simoncelli, E. P., Freeman, W. T., Adelson, E. H., & Heeger, D. J. (1992). Shiftable multiscale transforms. IEEE transactions on Information Theory, 38(2), 587-607.

[18] Heeger, D. J. (1998). Notes on steerable filters.

[19] Tartavel, G., Gousseau, Y., & Peyré, G. (2015). Variational texture synthesis with sparsity and spectrum constraints. Journal of Mathematical Imaging and Vision, 52, 124-144.

[20] Portilla, J., & Simoncelli, E. P. (2000). A parametric texture model based on joint statistics of complex wavelet coefficients. International journal of computer vision, 40, 49-70.

[21] Mémoli, F. (2007). On the use of Gromov-Hausdorff distances for shape comparison.

[22] Flamary, R., Courty, N., Gramfort, A., Alaya, M. Z., Boisbunon, A., Chambon, S., ... & Vayer, T. (2021). Pot: Python optimal transport. The Journal of Machine Learning Research, 22(1), 3571-3578.

[23] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE signal processing magazine, 29(6), 141-142.

[24] Solomon, J., De Goes, F., Peyré, G., Cuturi, M., Butscher, A., Nguyen, A., ... & Guibas, L. (2015). Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. ACM Transactions on Graphics (ToG), 34(4), 1-11.

[25] Vacher, J., Davila, A., Kohn, A., & Coen-Cagli, R. (2020). Texture interpolation for probing visual perception. Advances in neural information processing systems, 33, 22146-22157.

[26] Efros, A. A., & Freeman, W. T. (2023). Image quilting for texture synthesis and transfer. In Seminal Graphics Papers: Pushing the Boundaries, Volume 2 (pp. 571-576).

[27] Efros, A. A., & Leung, T. K. (1999, September). Texture synthesis by non-parametric sampling. In Proceedings of the seventh IEEE international conference on computer vision (Vol. 2, pp. 1033-1038). IEEE.

[28] Wei, L. Y., & Levoy, M. (2000, July). Fast texture synthesis using tree-structured vector quantization. In Proceedings of the 27th annual conference on Computer graphics and interactive techniques (pp. 479-488).

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

[30] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative adversarial nets. Advances in neural information processing systems, 27.

[31] Ho, J., Jain, A., Abbeel, P. (2020). Denoising diffusion probabilistic models. Advances in neural information processing systems, 33, 6840-6851.

[32] Arjovsky, M., Chintala, S., Bottou, L. (2017, July). Wasserstein generative adversarial networks. In International conference on machine learning (pp. 214-223). PMLR.

A SUPPLEMENTARY FIGURES

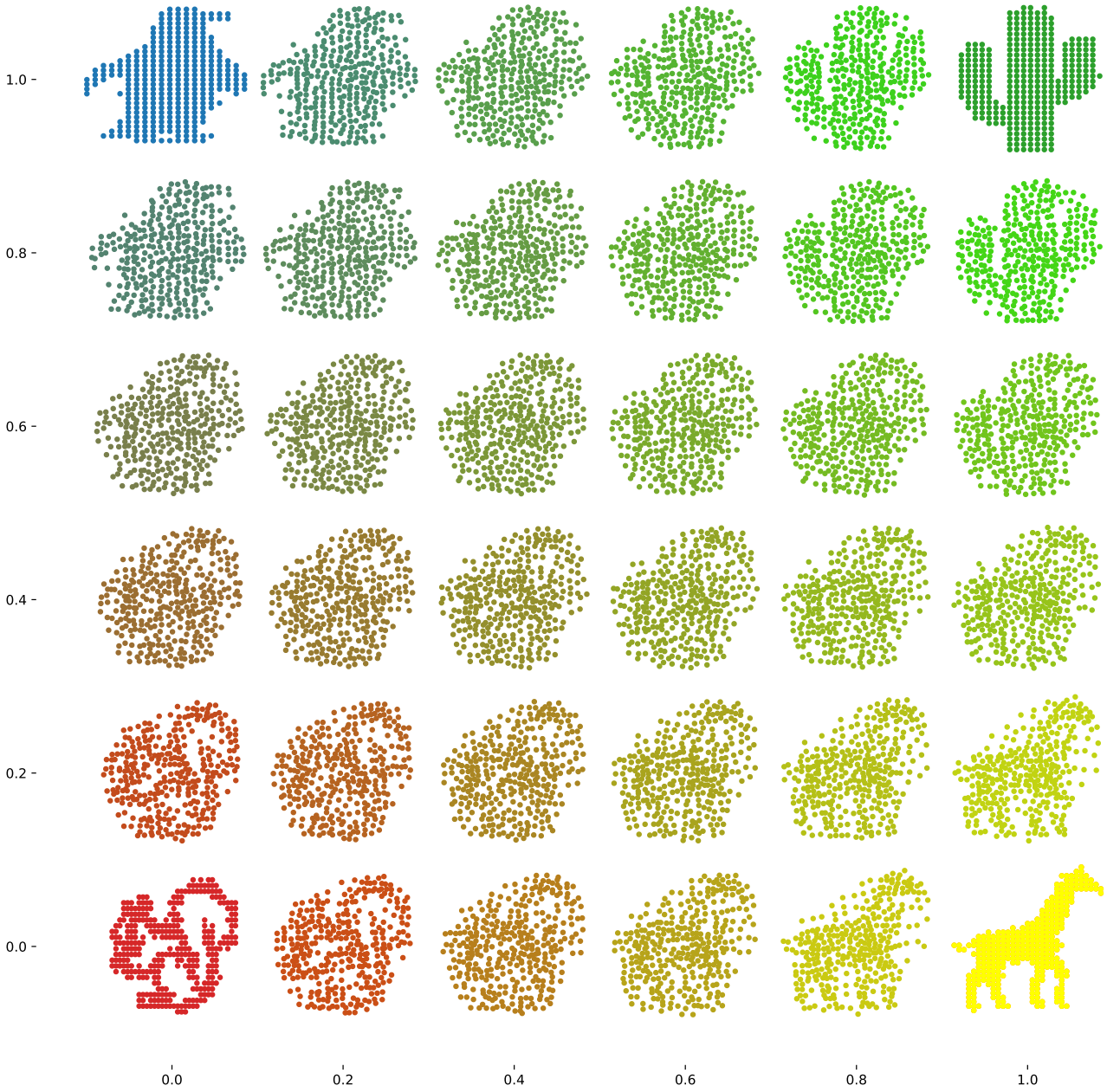
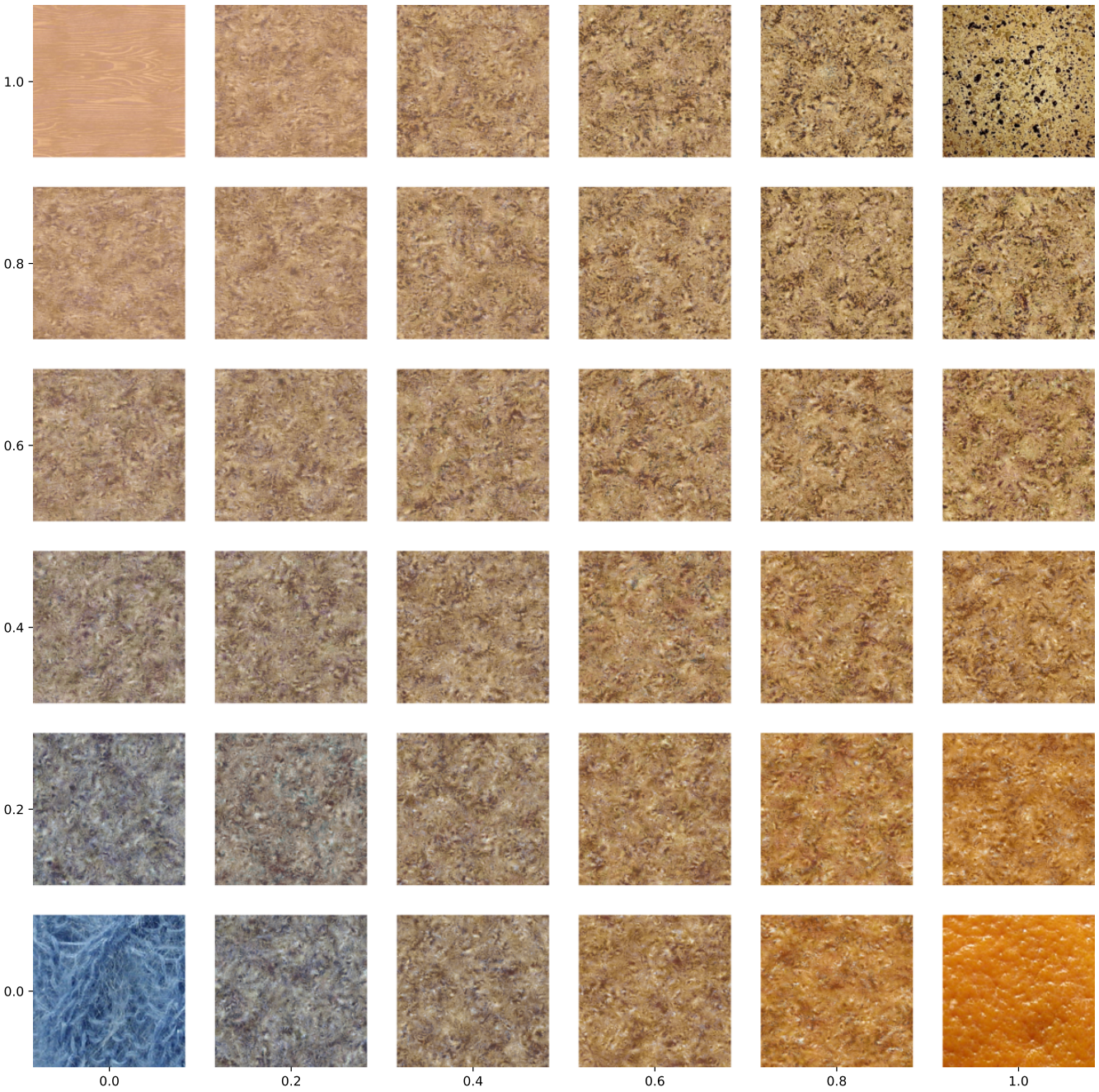
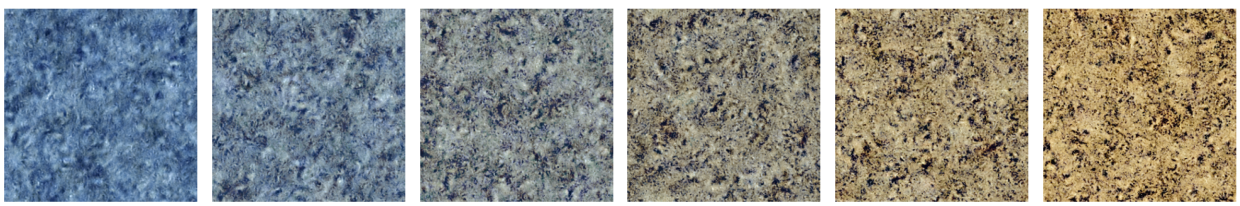


Figure 5: Sliced-Wasserstein barycenter between 4 point clouds

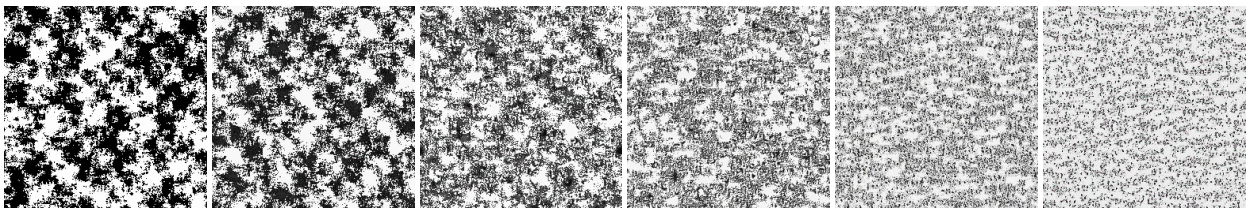


(a) Texture mixing between 4 textures



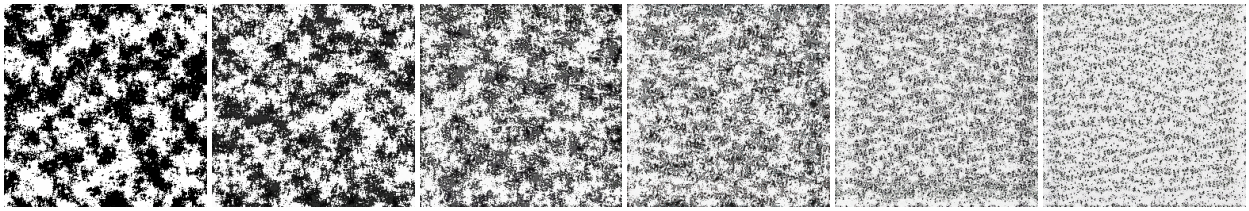
(b) Mixing between two textures

Figure 6: Texture mixing without any constraint



$\lambda = 0.0$ $\lambda = 0.2$ $\lambda = 0.4$ $\lambda = 0.6$ $\lambda = 0.8$ $\lambda = 1.0$

(a) Texture mixing with non-overlapping patches



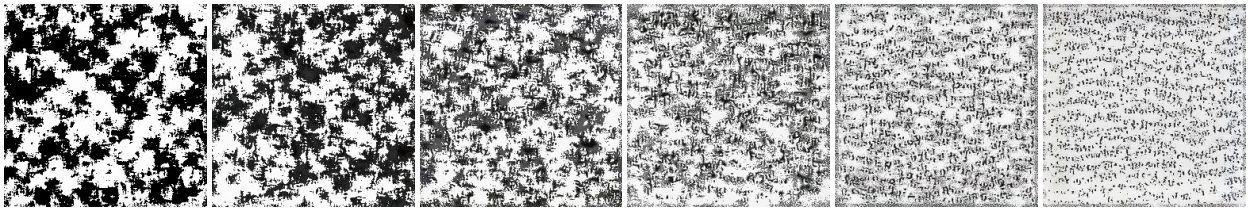
$\lambda = 0.0$ $\lambda = 0.2$ $\lambda = 0.4$ $\lambda = 0.6$ $\lambda = 0.8$ $\lambda = 1.0$

(b) Texture mixing with overlapping patches, $(2, 2)$ stride with mean on the overlaps

Figure 7: Texture synthesis with patch size = 4

$\lambda = 0.0$ $\lambda = 0.2$ $\lambda = 0.4$ $\lambda = 0.6$ $\lambda = 0.8$ $\lambda = 1.0$

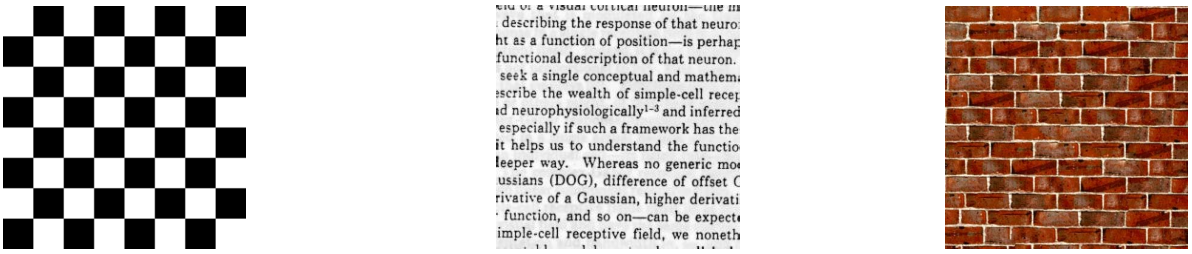
(a) Texture mixing with non-overlapping patches



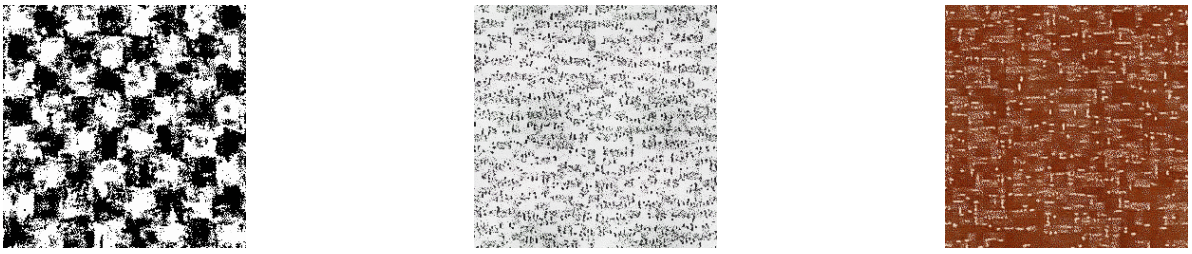
$\lambda = 0.0$ $\lambda = 0.2$ $\lambda = 0.4$ $\lambda = 0.6$ $\lambda = 0.8$ $\lambda = 1.0$

(b) Texture mixing with overlapping patches, $(4, 4)$ stride with mean on the overlaps

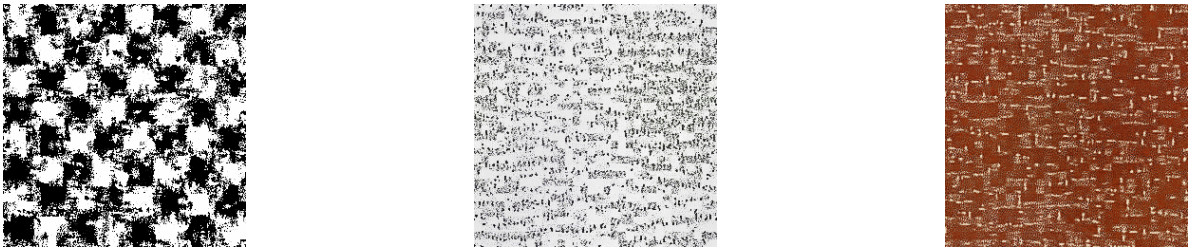
Figure 8: Texture synthesis with patch size = 8



Original images



$S = Q = 5$



$S = Q = 6$

Figure 9: Texture synthesis with patch size = 8, no overlapping with different values of S and Q

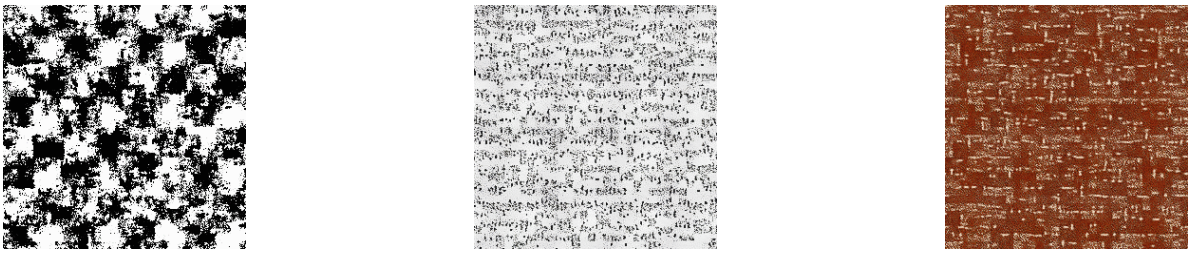


Figure 10: Texture synthesis with patch size = 8, no overlapping, and spectrum constraint

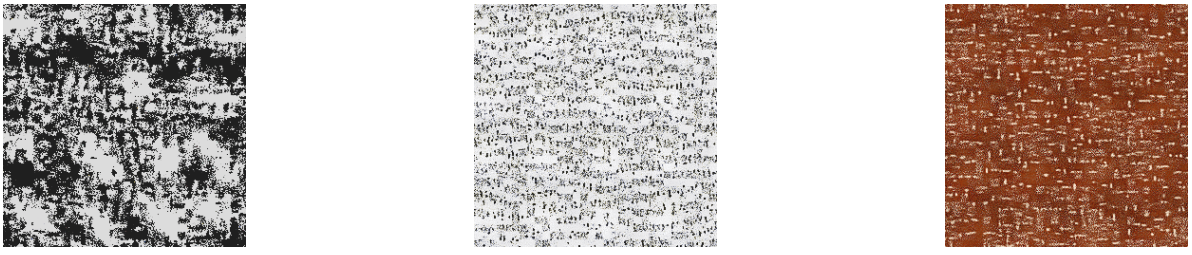


Figure 11: Texture synthesis with patch size = 8, no overlapping, no spectrum constraint and Gromov-Wasserstein minimization