NIFTY: a Non-Local Image Flow Matching for Texture Synthesis

Pierrick Chatillon, Julien Rabin, David Tschumperlé

NIFTY: a Non-Local Image Flow Matching for Texture Synthesis

Pierrick Chatillon, Julien Rabin, and David Tschumperlé

Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, F-14050 Caen, France

ABSTRACT

This paper addresses the problem of exemplar-based texture synthesis. We introduce NIFTY, a hybrid framework that combines recent insights on diffusion models trained with convolutional neural networks, and classical patch-based texture optimization techniques. NIFTY is a non-parametric flow-matching model built on non-local patch matching, which avoids the need for neural network training while alleviating common shortcomings of patch-based methods, such as poor initialization or visual artifacts. Experimental results demonstrate the effectiveness of the proposed approach compared to representative methods from the literature. Code is available at https://github.com/PierrickCh/Nifty.git.

***Index Terms*—** Generative model, Image synthesis, Texture synthesis, Flow Matching

1. INTRODUCTION AND RELATED WORK

Image generative modeling has been a very active domain over the past decade, driven by a combination of theoretical and technical advances. This progress has led to the development of diverse generative modeling frameworks, many of which rely on the training of deep neural networks.

Diffusion models Generative diffusion models (DMs) [1, 2, 3, 4, 5, 6, 7] have recently attracted significant attention for their ability to capture complex data distributions and generate high-quality samples, while benefiting from the stable training provided by conditional U-Net architectures. Large pre-trained DMs, often conditioned on text prompts [5], have also become widely used for inverse problems and image editing (*e.g.* [8]), and can be efficiently fine-tuned to address task-specific applications [9].

A major limitation of DMs lies in their sequential sampling process, formulated as a stochastic differential equation (SDE) [1, 2, 3], which requires a large number of small time steps during inference. This drawback has been alleviated by variants based on ordinary differential equations (ODE) [10], where stochasticity is restricted to the initialization, including implicit models such as DDIM [4], Rectified Flow (RF) [6], and Flow Matching (FM) [7]. Inference can be significantly accelerated by distillation-based approaches, in which a student network is trained to reduce the number of required steps [11, 6].

Texture Modeling with local patches Here we focus on exemplar-based texture synthesis, a task for which DMs are also gaining popularity, either through large pre-trained models [12] or models specifically trained on a single image [13]. Before the emergence of such approaches, stationary texture modeling had been studied through a variety of statistical techniques designed to learn local representation. A particularly influential framework relevant to our method is the seminal work of Kwatra *et al.* [14], which introduced patch-based texture optimization (TO) which has since inspired single-image generative models based on patch representations, *e.g.* [15, 16].

In [14], an image x is synthesized by minimizing an energy function defined w.r.t to a reference image u, where each patch of the synthesized image must closely match a patch from u. In practice, this non-convex energy minimization alternates between assigning each patch to its Nearest-Neighbor (NN) and averaging the overlapping patches at each pixel location. The method is, however, highly sensitive to random initialization and to hyper-parameters such as patch size and stride. When combined with a multi-scale strategy (coarse-to-fine synthesis), it can produce realistic samples, but often by replicating large regions of the reference image u, a behavior reminiscent of earlier methods that explicitly modeled such copy-paste effects [17]. These replicated regions are not always well aligned, frequently leading to discontinuities or blur, and fail to capture long-range correlations present in the reference. More recently, GAN-based approaches for single-image generation, such as [15], have been shown to exhibit the same limitations and can in fact be advantageously replaced by patch-based NN methods [18], without training a latent

This work was partly funded by the Normandy Region through the IArtist excellence label project and benefited from computational resources provided by CRIANN.

generative model.

Creativity of generative models Since then, the question of creativity in generative models trained on large datasets has been closely examined. Studies such as [19] have shown that these models are prone to memorization, and that some training data can be extracted. Recently, [20] demonstrated that an optimal diffusion models, *i.e.* maximizing the likelihood of a training set boils down to memorizing it. In practice, training a Convolutional Neural Networks (CNNs) to approximate the score driving the likelihood introduces a strong inductive bias: the training dataset is effectively processed as a collection of patches. This analysis allows for an explicit formula to model the learned diffusion by a CNN, where the score of a noisy sample at each time step essentially reduces to a mixture of Gaussians around each training patches. While this approach is not practical for large-scale applications, it further highlights the connection between patch-based NN matching and diffusion processes.

Contributions and Outline In this work, we propose NIFTY, an approximation of the explicit flow on patches for texture synthesis, that eliminates the need for neural network training and frame the TO algorithm [14] as a temporal integration of the flow (Sec. 2). In addition to coarse-to-fine synthesis with subsampling and aggregation strategies, we further introduce techniques, including top-k sampling and memorization, to reduce the number of required steps while maintaining quality. This approach offers several advantages shown in experiments (Sec. 3): robustness to initialization, image quality, and speed.

2. METHOD

The Flow Matching (FM) framework, as introduced in [7, 6] consists in sampling a complex data distribution by integrating a velocity field, starting from Gaussian samples. This velocity field is often approximated by a CNN. As discussed in the introduction, the fact that CNNs are used to compute the flow (or diffusion denoising steps) introduces the inductive bias of equivariance. As a consequence, computing the velocity field at a noisy image can be understood as applying the same local estimation to all patches. Subsequently, we will detail how to compute explicitly the flow on patch distributions (Sec. 2.1) and then approximate it efficiently (Sec. 2.2) without neural networks for texture synthesis.

2.1. Non-Local Image Flow-Matching

Notations and formal setting We consider the collection of training patches $\mathcal{P} := \{\phi\}$ from the reference image u with given patch size and stride. Note that to avoid border effect in practice, patches across boundaries are simply discarded.

We denote by $\mathcal{N}(\mu, \sigma^2)$ the isotropic multivariate Gaussian probability law with mean μ and standard deviation σ, and its density by $g_{\mu,\sigma}(x) = g_{0,1}(\frac{x-\mu}{\sigma}) \propto \exp -\frac{(x-\mu)^2}{2\sigma^2}$.

FM formulation Starting from a random (patch) sample $\psi_0 \sim \mathcal{N}(0, I)$ at time step $t = 0$, the FM framework consists during inference in solving the following ODE for $t \in (0, 1]$:

$$\partial_t \psi_t = v(\psi_t, t) \tag{1}$$

where v is the velocity of the flow such that generated samples $\psi_1 \sim \mathbb{U}(\mathcal{P})$ match the training distribution.

In [7], authors argue that while the problem of defining such a vector field is intractable, different vector fields are solution. They show however that the problem can be tackled by considering an affine parametrization of the conditional flow ψ_t given a target sample ϕ, using

$$\psi_t = t\phi + (1-t)\psi_0. \tag{2}$$

Training In practice, the corresponding vector field is learned using a neural network v_θ trained to minimize

$$\min_\theta \mathbb{E}_{\substack{\phi\sim\mathbb{U}(\mathcal{P}), z\sim\mathcal{N}(0,I) \\ t\sim\mathbb{U}([0,1])}} \|v_\theta(\phi_t, t) - (\phi - z)\|^2 \tag{3}$$

Interpretation Optimizing (3) on all possible functions at each time step is a barycentric problem which solution writes

$$v(\psi, t) = \mathbb{E}_{\substack{\phi\sim\mathbb{U}(\mathcal{P}) \\ z\sim\mathcal{N}(0,I)}} \Big(\phi - z \Big| \psi = \phi t + z(1-t)\Big). \tag{4}$$

Besides, the push-forward of the gaussian latent prior conditionally to a data point ϕ using such the affine mapping (2) is a conditional Gaussian probability distribution

$$p(\psi_t|\phi) = g_{t\phi,(1-t)^2}(\psi_t). \tag{5}$$

Closed-form solution Combining expressions (4) and (5) shows that one needs to sample a latent variable $z = \frac{\psi - t\phi}{1-t}$ to satisfy the condition $\psi = \phi t + z(1-t)$. Using the Bayes' rule, the computation of the conditional expectation (4) then reduces to a Gaussian mixture cen-

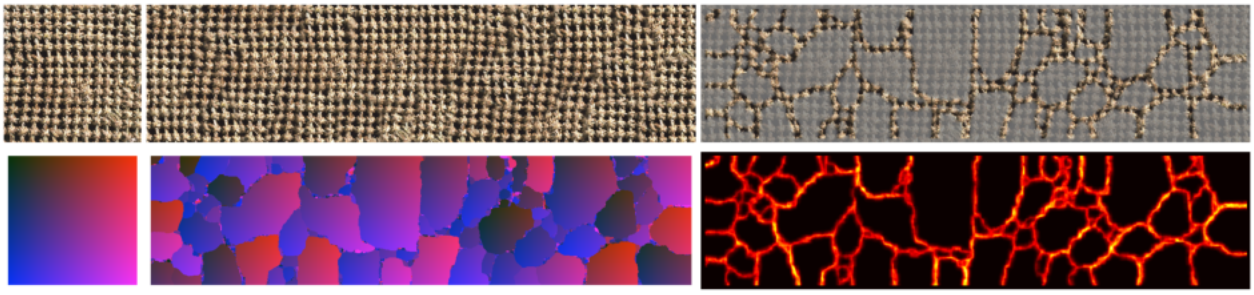

Fig. 1: **Creativity in synthesis.** Distance to the Nearest Neighbor (NN) patch in the training set is used to highlight novel structures. From left to right, first row: example image, generated sample, new structures; second row: pixel colormap, NN colormap, NN distance. This illustrates how the generative model manages to stitches local copies of the original texture by creating unseen yet likely structures.

tered around each training data point

$$\begin{aligned} v(\psi,t) &= \mathbb{E}_{\phi\sim\mathbb{U}(\mathcal{P})}\left(\tfrac{\phi-\psi}{1-t}\middle|\psi\right) \\ &= \frac{1}{1-t}\int_{\phi}(\phi-\psi)\frac{p(\psi|\phi)}{p(\psi)}dp(\phi) \\ &= \frac{1}{1-t}\frac{1}{|\mathcal{P}|p(\psi)}\sum_{\phi\in\mathcal{P}}(\phi-\psi)p(\psi|\phi) \\ &= \frac{1}{1-t}\sum_{\phi\in\mathcal{P}}(\phi-\psi)\omega_{\psi,t}(\phi) \\ \text{with } \omega_{\psi,t}(\phi) &= \frac{g_{t\phi,(1-t)^2}(\psi)}{\sum_{\phi'} g_{t\phi',(1-t)^2}(\psi)} \end{aligned} \tag{6}$$

Note that this formula is analogous to [20, Eq. 5] derived for an optimal equivariant score machine. Interestingly, this formula is also reminiscent of the Non-Local Means [21] where noisy patches are aggregated for image denoising applications.

When starting the flow at $t = 0$, it boils down to the direction towards the empirical mean: $\bar{\phi} - \psi$. Unfortunately, this formula becomes prohibitively expensive to compute for large numbers of patches, since all the dataset has to be used to compute the velocity at each step, for each patch under synthesis.

2.2. NIFTY Algorithm

The NIFTY algorithm, stands for Non-local Image Flow-matching Texture sYnthesis. It operates the previously introduced FM on patches in a *multi-scale* manner. At each resolution level, a noisy version of the upsampled signal from the previous level is denoised by solving the FM ODE (1) based on the proposed approximation (6). The computational cost is reduced through two main approximations.

Top-k NN Instead of evaluating the full weighted sum in equation (6), we restrict computation to the k patches $\{\hat{\phi}_i\}_{i\leq k}$ with the largest weights, which are the k-NN of $\frac{1}{t}\psi$. The corresponding weight approximation is given by:

$$\hat{\omega}_{\psi,t}(\hat{\phi}_j) = \frac{g_{t\hat{\phi}_j,(1-t)^2}(\psi)}{\sum_{i=1}^{k} g_{t\hat{\phi}_i,(1-t)^2}(\psi)} \tag{7}$$

Memory To further reduce computation, we consider only a subset $\{\phi^r\} \subset \mathcal{P}$ of the training patch set, where $|\{\phi^r\}| = r|\mathcal{P}|$ for some sampling ratio $r \in (0, 1]$. To mitigate the degradation of synthesis caused by this subsampling, we introduce a *memory function*: $m : [\![1, k]\!] \to [\![1, |\mathcal{P}|]\!]$, which retains the indices of the k-NN of each synthesized patch ψ across iterations t, only updated with better matches.

Pseudo-code Combining the two approximations, the flow for a patch ψ at time t and each scale is computed as follows:

1. **Sampling**: Select a subset $\{\phi^r\}$ of cardinality $r|\mathcal{P}|$.
2. **Neighborhood search**: Find $\{\hat{\phi}\}$, the k-NN of ψ within the set $\{\phi^r\} \cup \{\phi_{m(i)} \,|\, 1 \leq i \leq k\}$.
3. **Memory update**: Store new indices of the k-NN in m.
4. **Weight computation**: Use (7) to compute $\hat{\omega}_{\psi,t}(\hat{\phi}_j)$.
5. **Velocity estimation**: Compute the weighted velocity:

$$\hat{v}(\psi,t) = \frac{1}{1-t}\sum_{i=1}^{k}\left(\hat{\phi}_i - \psi\right)\hat{\omega}_{\psi,t}(\hat{\phi}_i) \tag{8}$$

6. **Flow update:** an Euler scheme is used to update the patch ψ using (8) with ODE (1);
7. **Stride:** to accelerate synthesis, we sample patches from the current synthesis with a spatial stride;
8. **Aggregation:** overlapping patches at a pixel are averaged using a spatial gaussian kernel.

Link between NIFTY and TO As recalled in Sec. 1, TO consist in alternating between patch-matching and aggregation, at different image resolutions and for different patch sizes (whereas we fix the patch size in NIFTY). As such, TO resembles to the proposed approach when selecting a single NN patch ($k = 1$) and performing a single time step ($T = 1$). Direct replacement with the nearest neighbor is a violent operation, which makes this algorithm sensible to initialization. By merging the flows, rather than the nearest matches, our approach has a more stable behavior and does not necessitate to go through different patch sizes at each resolution. As t tends to 1, the weights $\omega_{\psi,t}$ (which sum to 1) tends to the indicator of the nearest neighbor, and thus the flow points directly from ψ to its nearest neighbor. In that way, the final steps of our algorithm resemble the steps of TO.

It is worth noting that this non-local interpretation of FM for the TO algorithm is related to the work of [22], which also derives an explicit formula for DDPM. A key difference, however, is that their formulation restricts the score approximation to local patch NN matching, *i.e.*, comparing only patches at identical spatial locations across registered images.

3. EXPERIMENTS

We first present experimental results (comparison, ablation) of the proposed NIFTY approach using patch based representation. We consider in all experiments a fixed patch-size of 16 px with a sampling stride of 4 px, at 4 different scales. Recall that TO algorithm makes use of several patch size (32, 16, 8 px). Then, we show how the proposed framework can be easily transposed to latent presentations, *e.g.* latent DM [5].

Pixel-space NIFTY Figure 2 illustrates the effect of introducing memory in combination with top-k patch selection for improving the flow approximation. The flow quality is evaluated using the Wasserstein distance between patch distributions, computed on non-overlapping patches to avoid the influence of aggregation regularization. As expected, increasing k yields a better fidelity. Adding memory allows for comparable synthesis with a fraction of the computational cost.

Figure 1 shows an example of texture synthesis with the NIFTY algorithm. It illustrates how the proposed techniques drive the proposed FM to copy local regions of the training image (as demonstrated in [20]), while creating new unseen yet likely structures. A comparison in Fig. 3 to the TO algorithm underlines the interest of the proposed approach which avoid strong artifacts between copied regions.

To confirm this visual inspection, a quantitative comparison with TO is proposed in Table 1, computed from

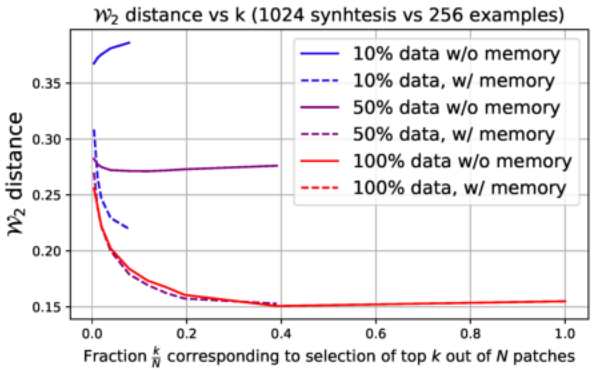

Fig. 2: **Effect of k-NN memory.** See text for details.

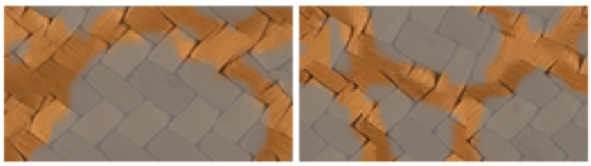

Fig. 3: **Artifacts.** Left: synthesis by NIFTY. Right: synthesis by TO. The synthesized patches not equal to their nearest neighbor from the reference are highlighted, as detailed in Fig.1.

10 synthesis (at 512 px res.) for 12 reference images (at 256 px). For reference, a small U-Net ($.7M$ parameters) is trained for each reference image to solve (3), highlighting the interest of the proposed approach in terms of quality and speed.

Method	Gram ↓ [23]	SIFID ↓ [15]	Autocorrelation ↓	Time (s)
NIFTY ($T = 15, k = 5$)	3 601	0.28	85.9	0.70
TO	12 676	0.76	431.8	1.92
U-Net ($T = 15$)	17034	0.54	133.1	600 (train) 0.13 (eval.)

Table 1: **Quantitative analysis.** Comparison using 3 metrics.

Figure 4 provides a visual comparison between image synthesis with NIFTY and with a U-Net approximation of the flow. Both synthesis are similar, and use the same random Gaussian initialization, demonstrating that they indeed approximate the same flow. To do so NIFTY is restricted to the finest scale for generation, degrading the results.

Latent-Nifty Figure 5 presents a synthesis result obtained by combining NIFTY with an autoencoder pretrained on the example patch distribution, highlighting the potential of the proposed method to be applied in latent spaces.

4. CONCLUSION

We introduced a generative model based on Non-Local Flow-Matching inspired from recent insights on deep

Fig. 4: **Mono-scale NIFTY vs FM network synthesis** Left: using our explicit approximation ($k = 10$). Right: using a U-net approximation of FM. Both methods start from the same noise realization.

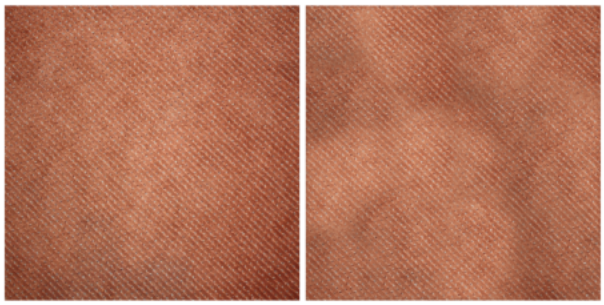

Fig. 5: **Latent NIFTY model** Left: reference image. Right: Our synthesis algorithm performed in the latent space of an auto-encoder

convolutional generative diffusion models. Revisiting the texture optimization algorithm from [14], we show that we can generate efficiently large random textures from a single example without the need to train a neural network, while avoiding the major pitfalls of the original approach, such as visual artefacts and sensitivity to initialization.

In addition to exploring patch flow matching in suitable latent spaces, future work includes the integration of more sophisticated inductive biases. For instance, attention modules could be modelled by non-local patch aggregation.

5. REFERENCES

[1] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *ICML*, 2015, pp. 2256–2265.

[2] Jonathan Ho, Ajay Jain, and Pieter Abbeel, "Denoising diffusion probabilistic models," *NeurIPS*, vol. 33, pp. 6840–6851, 2020.

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

[4] Jiaming Song, Chenlin Meng, and Stefano Ermon, "Denoising diffusion implicit models," in *International Conference on Learning Representations*, 2021.

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

[6] Xingchao Liu, Chengyue Gong, et al., "Flow straight and fast: Learning to generate and transfer data with rectified flow," in *11th International Conference on Learning Representations, ICLR 2023*, 2023.

[7] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le, "Flow matching for generative modeling," in *11th International Conference on Learning Representations, ICLR 2023*, 2023.

[8] Jiaming Song, Arash Vahdat, Morteza Mardani, and Jan Kautz, "Pseudoinverse-guided diffusion models for inverse problems," in *International Conference on Learning Representations*, 2023.

[9] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala, "Adding conditional control to text-to-image diffusion models," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 3836–3847.

[10] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine, "Elucidating the design space of diffusion-based generative models," *Advances in neural information processing systems*, vol. 35, pp. 26565–26577, 2022.

[11] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans, "On distillation of guided diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 14297–14306.

[12] Yang Zhou, Rongjun Xiao, Dani Lischinski, Daniel Cohen-Or, and Hui Huang, "Generating non-stationary textures using self-rectification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 7767–7776.

[13] Pierrick Chatillon, Julien Rabin, and David Tschumperlé, "Simulditex: A single image multiscale and lightweight diffusion model for texture synthesis," in *The 36th British Machine Vision Conference (BMVC)*, 2025.

[14] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra, "Texture optimization for example-based synthesis," in *ACM SIGGRAPH*, pp. 795–802. 2005.

[15] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli, "Singan: Learning a generative model from a single natural image," in *ICCV*, 2019, pp. 4570–4580.

[16] Antoine Houdard, Arthur Leclaire, Nicolas Papadakis, and Julien Rabin, "Wasserstein Generative Models for Patch-based Texture Synthesis," in *Scale Space and Variational Methods in Computer Vision*, Cabourg, France, May 2021, vol. LNCS 12679, pp. 269–280.

[17] Vivek Kwatra, Arno Schödl, and et al, "Graphcut textures: Image and video synthesis using graph cuts," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 277–286, 2003.

[18] Niv Granot, Ben Feinstein, Assaf Shocher, Shai Bagon, and Michal Irani, "Drop the gan: In defense of patches nearest neighbors as single image generative models," in *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 13460–13469.

[19] Ryan Webster and Teddy Furon, "Multi-modal identity extraction," in *ICCV 2025-International Conference on Computer Vision*, 2025.

[20] Mason Kamb and Surya Ganguli, "An analytic theory of creativity in convolutional diffusion models," in *International Conference on Machine Learning (ICML)*, 2025.

[21] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *Computer Vision and Pattern Recognition (CVPR)*, 2005, vol. 2, pp. 60–65.

[22] Denis Duval and Agnès Desolneux, "Réinterprétation des modèles génératifs de diffusion," in *30e Colloque sur le traitement du signal et des images*. 2025, pp. p. 1085–1088, GRETSI - Groupe de Recherche en Traitement du Signal et des Images.

[23] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge, "Texture synthesis using convolutional neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2015, p. 262–270.

A. DETAILED ALGORITHM

A more in-depth view of the NIFTY algorithm is described in Table 1.

Algorithm 1 NIFTY Patch Flow Matching

Input: Exemplar image u
Hyperparameters: Number of scales S, patch size p, number of neighbors k, renoising factor γ, subsampling ratio r, Number of timesteps T.
Output: Synthesized image

Initialize synthesis image $x \sim \mathcal{N}(0,1)$ with noise
$\{t_i\} \leftarrow T$ linear timesteps between 0 and 1
for $s = S-1$ to 0 **do**
  Resize u to resolution of scale s
  Extract training patches ϕ
  if $s > S-1$ **then**
    Upsample previous synthesis x
    Renoising: $x \leftarrow \gamma x + (1-\gamma)\varepsilon, \varepsilon \sim \mathcal{N}(0,1)$
    $\{t_i\} \leftarrow T$ linear timesteps between γ and 1
  end if
  for $i = 1$ to T **do**
    $\{\psi\} \leftarrow$ patches of x, using stride $\frac{p}{4}$
    Select $\{\phi^r\}$, uniformly sampling $\{\phi\}$
    Retrieve $\{\phi_{m(i)}, 1 \le i \le k\}$
    Compute $\hat{\phi}_{i\le k}$, the k-NN of $\frac{1}{t}\psi$ from $\{\phi^r\} \cup \{\phi_{m(i)}, 1 \le i \le k\}$, update m
    Compute weights $\hat{\omega}_{\psi_i^r,t}(\hat{\phi})$
    $\hat{v}(\psi,t) \leftarrow \frac{1}{1-t}\sum_{i=1}^{k}\big(\hat{\phi}_i - \psi\big)\hat{\omega}_{\psi,t}(\hat{\phi}_i)$
    $v_{agg}(x,t) \leftarrow$ Aggregation of $\hat{v}(\psi,t)$
    $\mathbf{x} \leftarrow \mathbf{x} + (t_{i+1} - t_i)v_{agg}(x,t)$
  end for
end for
return x

B. INTERPOLATION EXPERIMENTS

We display three approaches to blending texture exemplars using our patch-based method: distribution-level blending, pixel-level blending, and spatial interpolation of blending weights.

Distribution-level blending (Fig. 6) performs patch matching using the union of patches from both inputs. This yields an image with half of the locations looking like one exemplar.

Pixel-level blending (Fig. 7) instead computes patch matches separately for each input and interpolates their synthesis velocities linearly with a global α weight.

Finally, **spatial interpolation** (Fig. 8) uses a spatially-varying α map, enabling smooth transitions across regions and better spatial control over blending.

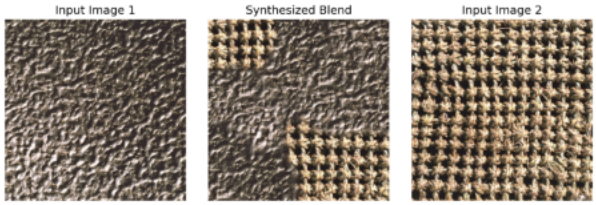

Fig. 6: Distribution-level blending. Patch matching is performed over the union of both input texture patch sets, enforcing statistical homogeneity.

Fig. 7: Pixel-level blending. The input textures (left, right) are synthesized by computing patch matches and interpolating their synthesis velocities with a global blending parameter $\alpha = 0.5$.

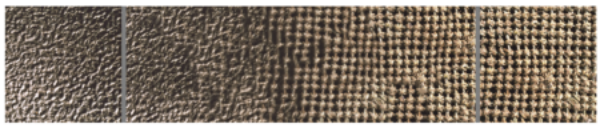

Fig. 8: Spatial interpolation. A spatially-varying α map is used to transition smoothly between input textures, enabling region-aware synthesis.

C. MORE RESULTS

For all methods and for 4 images among the 12 of the base, we show in Fig. 9,10 and 11 one of the 10 synthesis used to compute the values in Table 1

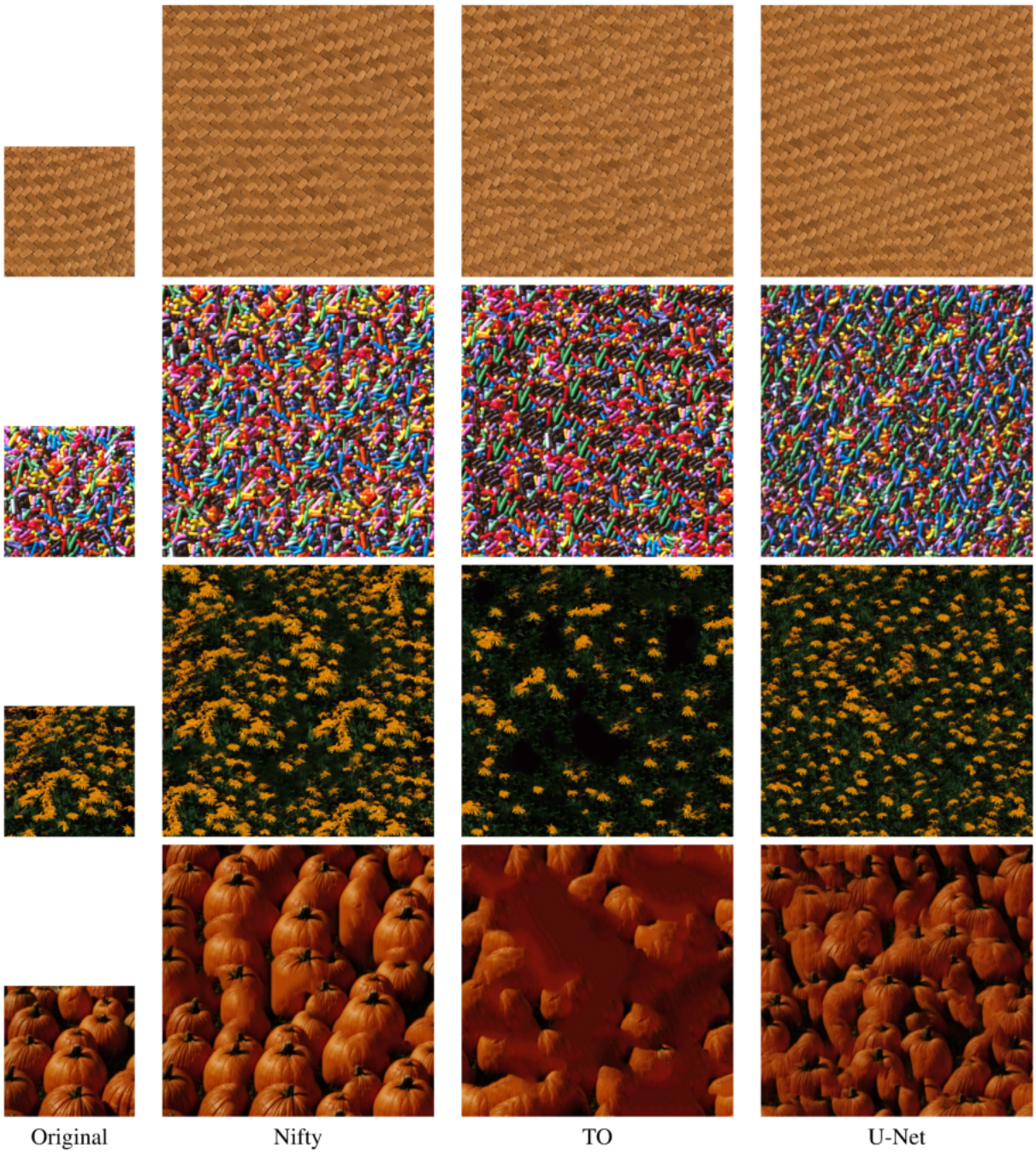

Fig. 9: Comparison of synthesis across methods and images from the dataset used to compute average score in Table 1.

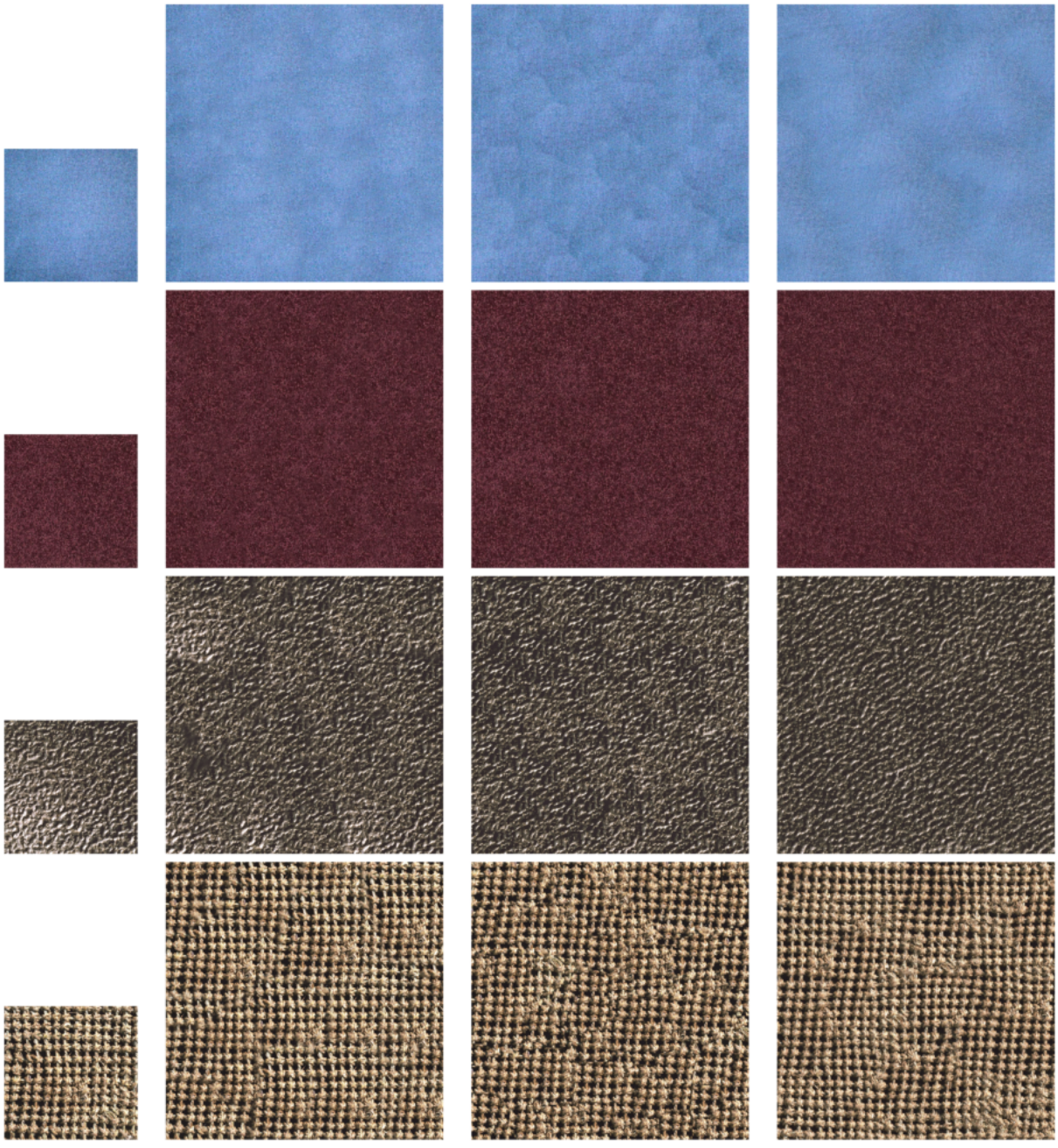

Fig. 10: Comparison of synthesis across methods and images from the dataset used to compute average score in Table 1.

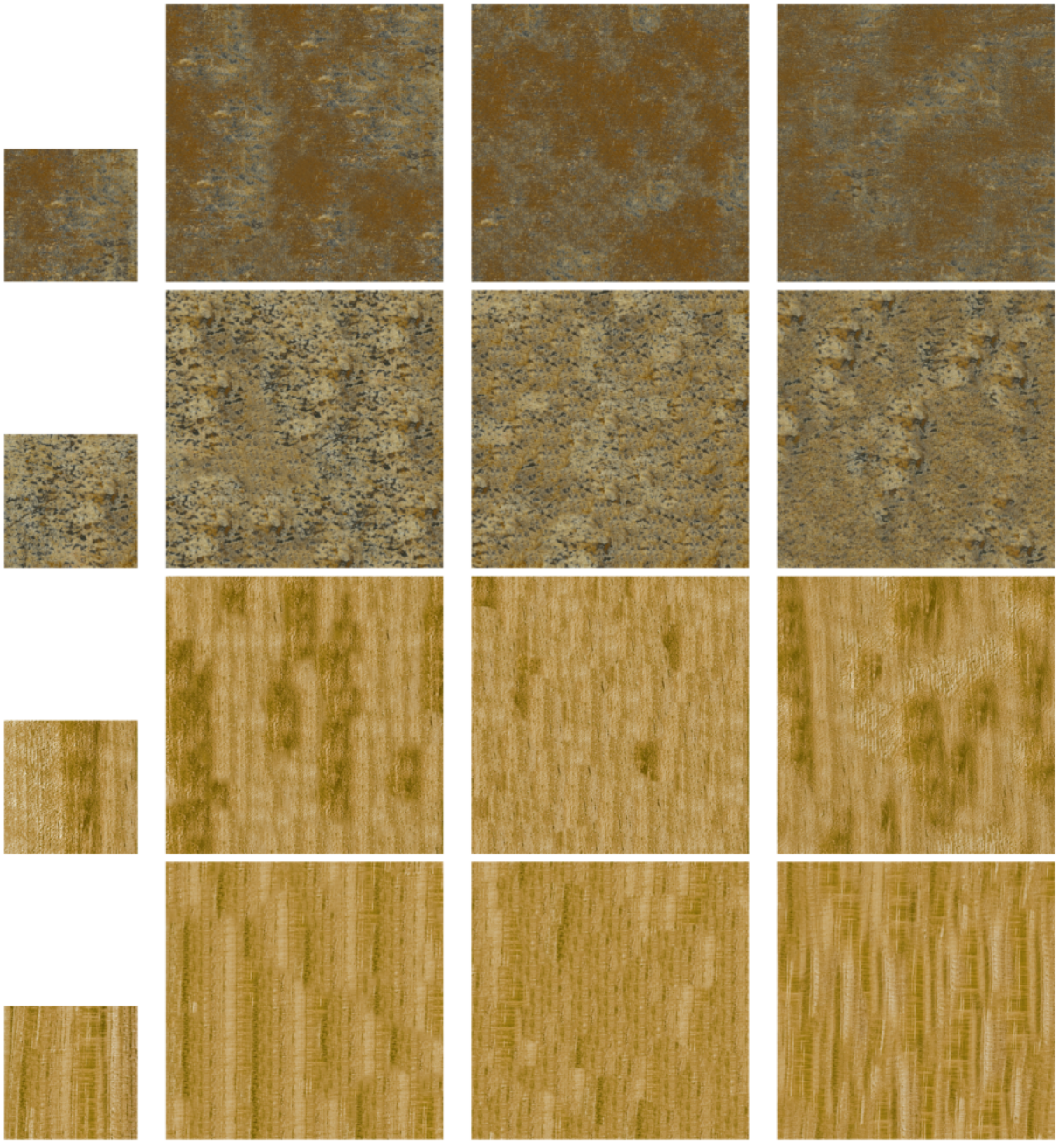

Fig. 11: Comparison of synthesis across methods and images from the dataset used to compute average score in Table 1.