Neural Texture Synthesis with Guided Correspondence

Yang Zhou Kaijian Chen Rongjun Xiao Hui Huang*
Visual Computing Research Center, Shenzhen University
{zhouyangvcc, szchenkaijian, xiaorongjun000, hhzhiyan}@gmail.com

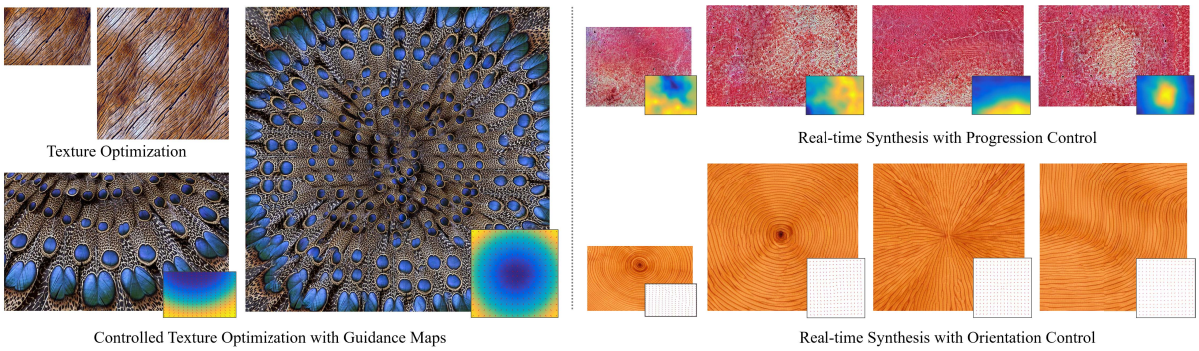

Figure 1. We propose an improved CNNMRF model. It can be used for example-based texture optimization with or without guidance maps (left). The newly defined loss function can also be applied to train generative networks, enabling real-time controlled synthesis (right).

Abstract

Markov random fields (MRFs) are the cornerstone of classical approaches to example-based texture synthesis. Yet, it is not fully valued in the deep learning era. This paper aims to re-promote the combination of MRFs and neural networks, i.e., the CNNMRF model, for texture synthesis, with two key observations made. We first propose to compute the Guided Correspondence Distance *in the nearest neighbor search, based on which a* Guided Correspondence loss *is defined to measure the similarity of the output texture to the example. Experiments show that our approach surpasses existing neural approaches in uncontrolled and controlled texture synthesis. More importantly, the Guided Correspondence loss can function as a general textural loss in, e.g., training generative networks for real-time controlled synthesis and inversion-based single-image editing. In contrast, existing textural losses, such as the Sliced Wasserstein loss, cannot work on these challenging tasks.*

1. Introduction

Example-based texture synthesis has been a long-standing topic in vision and graphics. It aims to synthesize new textures of any resolution that retain the patterns of a given exemplar, with no apparent visual flaws and having realism. Classical approaches formulate the synthesis as a Markov Random Field (MRF) problem and solve it by iteratively optimizing the output patches to be similar to their nearest neighbor in the input. This MRF-based optimization framework is not only widely used both in texture synthesis [18–20, 25, 39], but also adopted in more general tasks such as image synthesis and editing [1, 7].

Despite the success of MRF optimization, recent attention has been devoted to utilizing deep neural networks, either matching the statistics of deep features [12, 16] or training generative adversarial networks (GANs) [5, 30, 33, 40]. In this paper, we retake the MRF optimization framework, given its versatility and flexibility in texture synthesis, and combine it with deep neural networks. We first search the nearest neighbor for each output patch according to *Guided Correspondence Distance* over multi-layer deep features. Then, unlike traditional methods that copy and paste source patches, we define a *Guided Correspondence loss* that measures the overall similarity based on all the corresponding patches, and update the output pixels via back-propagation.

Actually, Li *et al*. [22] used to explore a CNNMRF model in 2016, which combines MRF and neural networks for style transfer. Champandard [6] applied it to texture synthesis later. Comparing to traditional texture optimization,

*Corresponding author

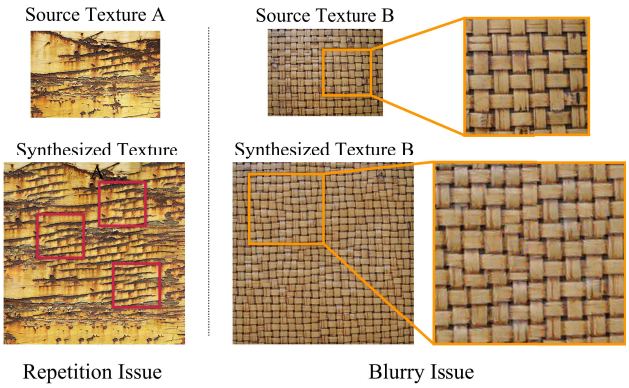

Figure 2. The synthesized textures from CNNMRF model [6] have obvious repetition and blurry issues.

the main issue of CNNMRF is that its results have a poor patch diversity and severe blurry artifacts; see, e.g., Figure 2. To address that, we made two critical changes in our approach. First, the Guided Correspondence Distance is defined as a weighted sum of various penalty terms. Thus, in the nearest neighbor search, we can take more factors such as matching diversity into account rather than only consider patch similarity. Second, inspired by the Contextual loss [26] used for matching image statistics in style transfer, we modify the conventional L_2-based MRF energy to account for contextual similarities. The motivation is that we hope the nearest neighbor we found for a target patch is significantly closer to it than all other source patches. The so-designed Guided Correspondence loss improves the sharpness of the synthesized results substantially. Our framework can be easily extended to various guided scenarios, including (but not limited to) user annotations, progression maps, and orientation fields. We just add the corresponding penalties to the Guided Correspondence Distance.

Experiments show that our approach performs remarkably well for texture optimization both in uncontrolled and controlled scenarios, reaching state-of-the-art visual quality. Moreover, the Guided Correspondence loss can be used as a general textural loss. We demonstrate its usage in, e.g., training feedforward networks for real-time controlled synthesis and inversion-based single-image editing. Existing statistic-based losses, such as the Sliced Wasserstein loss, cannot handle these challenging tasks. Code is available at https://github.com/EliotChenKJ/Guided-Correspondence-Loss.

2. Related Work

Classical approaches. Classical example-based texture synthesis approaches have gone through three stages: pixel-based [9, 37], stitching-based [8, 21], and optimization-based [20, 38]. Given the success of texture optimization, follow-up researches focus on such as PatchMatch acceleration [1, 7], self-guided optimization [18], and applications in various controlled scenarios [3, 24, 25, 29, 35, 39].

Deep learning based approaches. The first work using neural networks for texture synthesis is proposed by Gatys *et al*. [12]. They synthesize new textures by matching the correlations between deep features to be similar to the source. Heitz *et al*. [16] also leverage deep features but choose to align them between images by minimizing the Sliced Wasserstein Distance, resulting in better visual quality. They assume a strong prior of stationary statistics for textures and emphasize matching the distribution in the feature space while "overlooking" the spatial coherence on the target image. When an exemplar has large-scale structures or non-stationary changes in the spatial domain, matching global statistics will cause discontinuities or incoherent color distributions in the synthesized textures.

Another direction utilizing deep techniques is training generative networks [4,5,11,17,23,28,31,32,34]. Recently, several single-image GANs are proposed to learn the internal patch distribution [30,33,40]. They can reasonably deal with challenging non-stationary textures at the cost of long-time training but always suffer from visual artifacts, especially near the border areas. Another critical problem is that GANs have no explicit measure about the synthesis quality with respect to the exemplar. We may not know when to stop training to prevent overfitting.

3. Method

We start by describing the classical MRF global objective defined in [20] that measures the overall similarity between two images. The key idea behind MRF prior is to emphasize the visual coherence across all overlapping output patches. Formally, let I_t denote the output/target texture to be optimized, and I_s denote the input/source example. For controlled synthesis, there are also a certain target guidance map G_t and a corresponding source guidance map G_s. We represent each texture as a collection of overlapping patch samples: $T = \{t_i\}$ and $S = \{s_j\}$. $n_t = |T|$ and $n_s = |S|$ are the patch numbers. Usually, we assume $n_t > n_s$, as we always synthesize textures larger than the exemplar. The global MRF energy that measures the similarity of I_t to I_s can be written as:

$$E_{MRF}(I_t, I_s) = \sum_{t_i \in T} p(t_i, NN(i)), \tag{1}$$

where $NN(i) \in S$ is the nearest neighbor of t_i, and $p(\cdot,\cdot)$ is a similarity function, which is usually the sum of squared color distance in classical approaches. Classical approaches solve the MRF objective above by an Expectation-Maximization (EM)-like algorithm which iterates between updating the output pixels (E-step) and finding the nearest neighborhoods in input (M-step). In our framework, we will

convert the above energy into a loss function so that it can be used to optimize target pixels via back-propagation, or to update network parameters as a general textural loss.

Next, we will introduce the Guided Correspondence Distance we used in nearest neighbor search, followed by the detailed definition of Guided Correspondence loss.

3.1. Guided Correspondence Distance

Feature Representation. We extract multi-layer deep features for texture images using a pre-trained VGG-19, and select L layers from them to compute the patch distance for nearest neighbor search. Note that in controlled synthesis, the features in guidance maps should also be involved. Thus, the feature vector of a target patch t_i is comprised by $\{F_t^l(i), G_t^l(i)\}$, where F_t^l denotes the VGG activation at the l-th layer, G_t^l is the down-sampled target guidance map with the same size as F_t^l, and i represents the patch sampling on the feature map. For a source patch s_j, it is $\{F_s^l(j), G_s^l(j)\}$ with the same meaning of notations. Since we compute identically for all selected layers, we will ignore the superscript l in the following for simplicity.

Distance Metric. We formulate the pair-wise guided distance d_{ij} between patches t_i and s_j at a certain layer as:

$$d_{ij} = d_{ij}^{VGG} + \lambda_{GC} * d_{ij}^{GC} + \lambda_{occ} * d_j^{occ}, \qquad (2)$$

where d_{ij}^{VGG} is the cosine distance between neural patches $F_t(i)$ and $F_s(j)$, d_{ij}^{GC} is the distance between guidance patches $G_t(i)$ and $G_s(j)$, d_j^{occ} is the occurrence penalty (to be defined later) for s_j, and $\lambda_{GC}, \lambda_{occ}$ are the weights. The definition of distance function d_{ij}^{GC} depends on the type of guidance channel in controlled synthesis. We will specify it case by case in the experiment section. While for uncontrolled synthesis, we directly set $\lambda_{GC} = 0$.

Occurrence Penalty. Finding correspondences for target samples only by the features from image and guidance channel will cause repetition issues, as there's no guarantee to avoid the many-to-one degradation in matching. Inspired by the occurrence penalty used in self-tuning texture optimization [18], we implement a simplified version to encourage selecting source patches uniformly. Specifically, we first use the feature distance terms d_{ij}^{VGG} and d_{ij}^{GC} in Eq. (2) to get an approximated guided correspondence for each target patch. Then, for each source patch s_j, we count the number of times it is chosen as the guided correspondence and define its occurrence penalty as:

$$d_j^{occ} = \Omega_j / \omega, \qquad (3)$$

where $\Omega_j = |\{s_j = NN(t_i), \forall i\}|$, and $\omega = n_t/n_s$ is a normalization defined by the average occurrence number over all source patches.

$d_{ij}^{VGG} = 1 - \frac{(F_t(i)-\mu_s)\cdot(F_s(j)-\mu_s)}{\|F_t(i)-\mu_s\|_2 \cdot \|F_s(j)-\mu_s\|_2}$, where $\mu_s = \frac{1}{n_s}\sum_j F_s(j)$

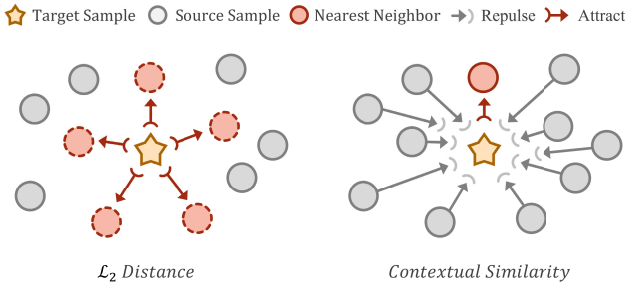

Figure 3. Illustration of our contextual similarity. Given the strong self-similarity of texture, there could be multiple source samples with similar distances to the same target sample. Minimizing L_2 distance makes the optimization favor the average of nearby source samples, resulting in blurry artifacts in the output. The contextual similarity, instead, considers the context of all source samples. A target sample is contextually similar to a source sample if it is significantly closer to it than to all other source samples. Minimizing the contextual similarity will push the target sample to get closer and closer to its nearest neighbor.

After updating the pair-wise distances with the occurrence penalty, we find the guided correspondence for each target patch. Next, we convert the Guided Correspondence Distance into Guided Correspondence loss.

3.2. Guided Correspondence loss

We could have directly used the distance metric of Eq. (2) as the similarity function $p(\cdot,\cdot)$ in the global objective. But we noticed severe blurry artifacts in the generated results. Such blurry issue can also be evidenced in the results of the original CNNMRF model that optimizes the L_2 distance of deep features between corresponding patches. We argue that this issue can be attributed to the strong self-similarity of textures. For a target sample (i.e., a patch), there could be multiple source samples with similar distances to it in feature space. Minimizing the Euclidean distance to its nearest neighbor would make the optimization stay or vibrate at the average of nearby source samples and hence output blurry patches. We, therefore, turn to the contextual similarity introduced in [26]. As illustrated in Figure 3, the contextual similarity requires a target sample to be significantly closer to its nearest neighbor than to all other source samples. To capture that, we first normalize the guided distance of Eq. (2) and shift it to similarity:

$$w_{ij} = \exp\left(\frac{1 - d_{ij}/(\min_k d_{ik} + \epsilon)}{h}\right), \qquad (4)$$

where $\epsilon = 1e-5$ is to prevent division by zero, and h is a bandwidth parameter which we set as 0.5 in all our experiments. Then, we normalize it to be the contextual similarity:

$$CX_{ij} = w_{ij} / \sum_k w_{ik}. \qquad (5)$$

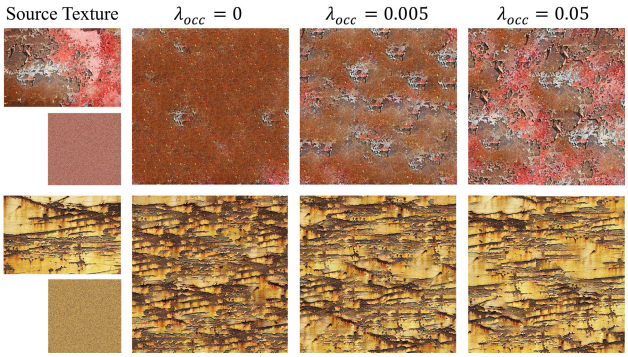

Figure 4. Ablation study on occurrence penalty. Note the results of all these three settings are optimized from the same random initialization (shown at the bottom left of each row).

Finally, we modify the MRF energy function of Eq. (1) and formally define our Guided Correspondence loss:

$$\mathcal{L}_{GC}(I_t, I_s) = \frac{1}{n_t}\sum_i -log(CX_{i,NN(i)}). \quad (6)$$

Note that our Guided Correspondence loss is different from the Contextual loss defined in [26]; please refer to the supplementary for more details. In texture optimization, we minimize the above loss by iterating between computation of guided correspondence and error back propagation so as to modify output pixels. While in training generative models, the above loss can be used to measure the quality of generated images so as to update network weights.

3.3. Implementation details

We select the feature layers $relu1\_1$, $relu2\_1$, $relu3\_1$, and $relu4\_1$ of VGG-19 to sample the neural patches with patch size 7 and stride 3 for the computation of Guided Correspondence loss. Following [22] and classical approaches, we use the multi-resolution strategy to further enlarge the receptive field in optimization-based synthesis. Specifically, we set 4 levels of resolution by default, which are [0.25, 0.5, 0.75, 1]. We run 500 EM iterations for each scale. It takes about 5 minutes to optimize a texture of 512 $\times$ 512 pixels with an Nvidia Titan Xp GPU.

In controlled scenarios, to meet the requirement of guidance channels, classical approaches usually apply generalized PatchMatch [2] to search for rotated patches. In our implementation, we directly augment the source patches with flipped and rotated copies. We found this simple augmentation significantly improves the synthesis quality. To save the memory cost, we remove layer $relu1\_1$ in the controlled synthesis. When there are 4 augmented copies (horizontal+vertical flips), the computation time is around 5 minutes for an output of 512^2 pixels, and 20 minutes when there are 8 augmented copies (each rotated by $45°$).

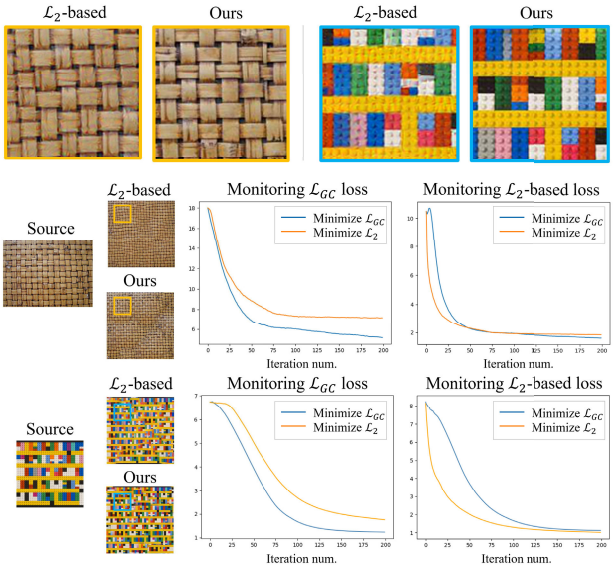

Figure 5. Ablation study on the contextual similarity in our loss function. By keeping all the other settings unchanged, we replace the contextual similarity in the Guided Correspondence loss with L_2 distance. Top: zoom-in patches in the synthesis results. Bottom: we monitor the evolution of the two losses during the optimization (at scale 0.25) respectively using either loss. The loss curves indicate that minimizing the contextual similarity based loss, i.e., the Guided Correspondence loss, minimizes the L_2 distances simultaneously, but it's not true for the opposite.

Table 1. We check the average number of correspondence switches for all target patches during the optimization (at scales 0.75 and 1, since now the target texture is optimized a lot). We can see using the contextual similarity based loss is almost two-times stable than using the L_2 distance in the nearest search, which hence leads to better optimization for the final output patches.

	Texture 1 in Fig. 5		Texture 2 in Fig. 5	
	L2-based	Ours	L2-based	Ours
Scale_0.75	14.09±5.33	**6.83**±4.12	8.01±5.91	**3.93**±3.93
Scale_1	2.43±4.17	**1.32**±3.21	1.39±3.97	**0.62**±2.14

4. Experiments

We select 50 texture images from [40] for experiments. Each is resized to 256 pixels for the shorter dimension. Detailed configurations of all experiments and more results in high resolution are included in the supplementary material.

4.1. Ablation study

We start by validating the two key components of our approach: the occurrence penalty in the Guided Correspondence search and the contextual similarity in our loss function. For the former one, we run our method for texture optimization and tune the penalty weight to see the effect.

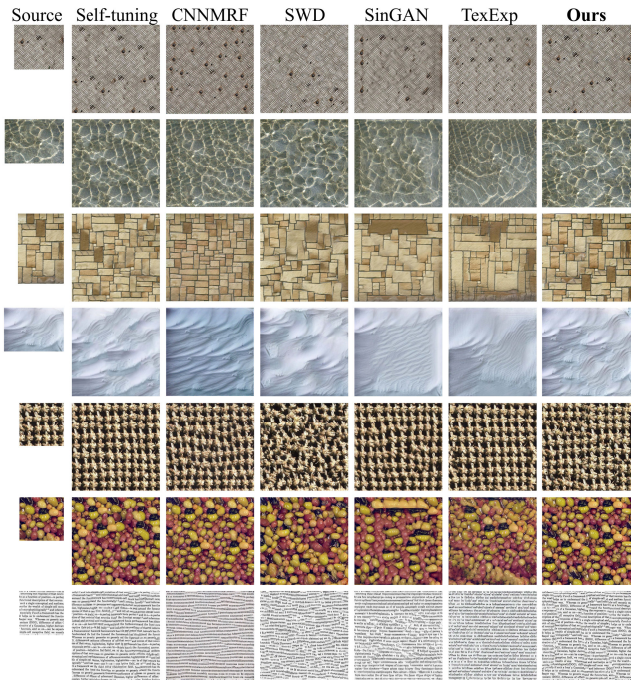

Figure 6. Qualitative comparison to state-of-the-art approaches on uncontrolled synthesis of stationary textures. All the resulting textures are in size 512×512 pixels. Note that TexExp can only output textures two times larger than the input. To get a square output, we expand the input first and then crop the central part. We did the same operation for SWD, avoiding unnecessary upsampling on the source patches according to the authors' suggestion. Please check the supplementary for more comparisons in high-resolution.

As shown in Figure 4, severe repetitions can be seen in the output when excluding the occurrence penalty. After raising its weight, the synthesized textures become more and more natural-looking. We fix the weight of occurrence penalty as 0.05 for most experiments unless specified.

Figure 5 shows the comparison between using the contextual similarity with using the L_2 distance in the Guided Correspondence loss. The visual comparison may be subtle to tell the difference. But the loss curves indicate that optimizing the contextual similarity based loss also minimizes the L_2 distance between correspondences, whereas the opposite is not true. Besides to track the loss curve, we further check the correspondence switches during the optimization. For the two textures shown in Figure 5, the statistics shown in Table 1 indicate that our contextual similarity based loss is much more stable in correspondence search than using the L_2 distance, which explains why our loss leads to better optimization for the target pixels.

4.2. Uncontrolled synthesis

The basic use of our method is running it for texture optimization. Figure 6 shows a gallery of uncontrolled synthesis results produced by our method along with qualitative comparison to results from several existing approaches, including: Self-tuning texture optimization [18], the state-of-the art classical method; CNNMRF [22], the baseline model of our method; Sliced Wasserstein loss (SWD) [16], the state-of-the-art statistic-based textural loss; SinGAN [30] and Texture Expansion networks (TexExp) [40], two state-of-the-art single-image GANs. Compared to CNNMRF, we can see that our results are consistently sharper and have fewer artifacts and, more importantly, fewer repetitions. While SWD deals well with stochastic textures, it struggles for textures with structures or large-scale texture elements; e.g., we may notice many broken lines and distorted patterns in rows 3, 5, and 7 of Figure 6. This is because SWD does not consider the local coherence between target samples. As for SinGAN and TexExp, they both have border issues and seemly fall into overfitting for some of the textures. In contrast, Self-tuning produces plausible results for almost all examples. Our results are visually comparable to theirs and only differ in some details. More results and comparisons are contained in the supplementary.

Table 2. Quantitative comparison on uncontrolled synthesis. We compute the Average Color Distance as the metric of synthesis quality and investigate the user preference over 40 participants with 2k questions. Note that 50% means comparable.

	Self-tuning	CNNMRF	SWD	SinGAN	TexExp	Ours
ColorDis	2.65	23.16	24.21	15.72	25.32	9.40
User Pref.	47.3%	33.1%	31.5%	9.57%	33.0%	-

Although there's no canonical metric to evaluate the synthesis quality, we still conducted two quantitative comparisons. First, we compute the Average Color Distance between the synthesized textures and the examples. We divide the output into tiles of size 10×10 pixels. For each tile we search exhaustively its nearest neighbor in the source by the sum of squared color distance. The average color distance of output tiles to their nearest neighbor is highly correlated with the synthesis quality. As shown in Table 2, Self-tuning has the minimal average color distance since it directly copies source patches to composite the target texture. Our results also have small color distance, especially given the significant improvement over CNNMRF, demonstrating the sharpness advance and artifacts reduction again.

We then did a user study to examine the human perception of synthesis quality. Each time we show the participant a random exemplar and two results: one from ours and the other from the competing methods. The user is asked to choose the better one or "comparable" otherwise. Each participant needs to finish 50 questions (10 for each competing method). 40 participants have been involved. We can see from Table 2 that our method wins the comparisons against the neural methods by a large margin, and performs slightly

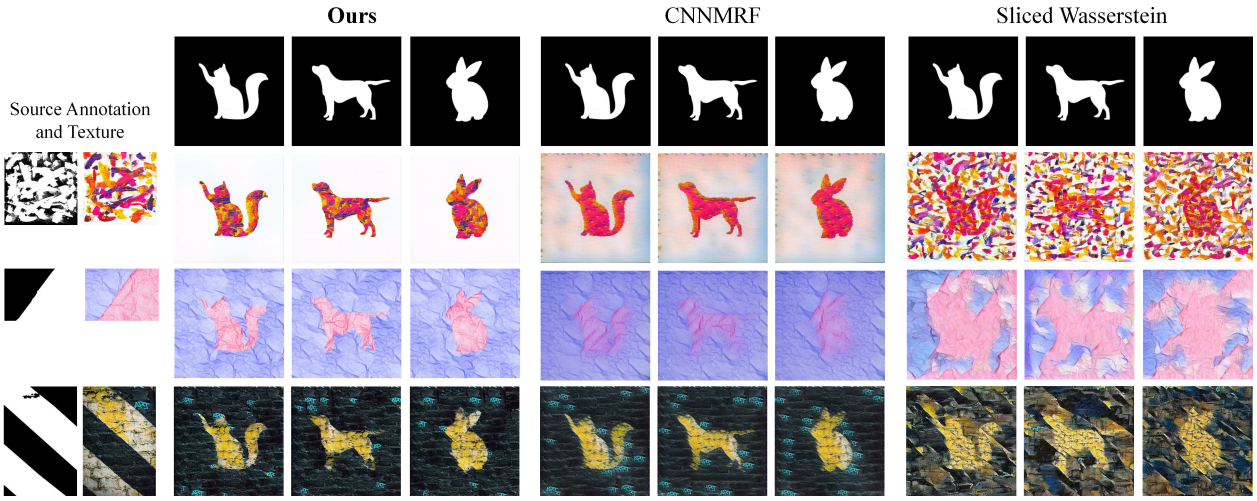

Figure 7. Our method can control the spatial distribution of textures according to user-specified annotation maps. CNNMRF [6] can do the same job but produces much worse visual quality. The Sliced Wasserstein loss [16], however, struggles on this task as it requires a strong bi-directional similarity between the source and target images, which may conflict with the controls specified by the user.

better than Self-tuning. However, Self-tuning uses a smart initialization and an additional edge map as the guidance for more coherent structures. Our method optimizes from random noise without edge guidance involved.

4.3. Controlled synthesis

Our approach is naturally suitable to cooperate with various guidance channels for controlled synthesis. We experiment with several types of guidance to demonstrate the flexibility and controllability of our method.

Annotation control. First is the annotation map for spatial control. A source label map is augmented with the source texture to characterize the element classes. The output layout is specified by a target label map. To achieve spatial control, we set d_{ij}^{GC} in Eq. (2) as the sum of pixel-wise L_0 distance between the patches on annotation maps:

$$d_{ij}^{GC} = \frac{1}{k*k}\sum_{x=0}^{k*k} ||G_t(j)(x) - G_s(i)(x)||_0, \qquad (7)$$

where k is patch size (7 in default). $G_s(i)(x)$ and $G_t(j)(x)$ denote the corresponding pixels in the patch on annotation maps. The weight λ_{GC} is set as 10.

As shown in Figure 7, our method precisely controls the output distribution and plausibly preserves the exemplar's fine details. CNNMRF realizes spatial control by concatenating the annotation maps to deep features before distance computation. Although controlled correctly, it produces less appealing results with severe repetition and blurry artifacts. The Sliced Wasserstein loss [16] also concatenates user annotations to deep features as spatial tags. To ensure a correct grouping of samples in feature space, these tags are set strictly larger than other dimensions. However, SWD cannot control the spatial distribution as expected. The reason is SWD enforces a strong bi-directional similarity, which may conflict with the requirements of user annotations as the element portions could be very different between the source and target label maps.

Progression and orientation control. Then we experiment with two automatically extracted source guidance channels proposed in [39]. One is a scalar progression map to control the texture distribution continuously. The other is a direction field to control the local dominant orientation of texture patterns. As shown in Figure 8, our method can achieve these two controls separately, or simultaneously. For separate control, we realize progression control by setting d_{ij}^{GC} in Eq. (2) as the sum of squared L_2 distance between patches on the progression maps, i.e.,

$$d_{ij}^{GC} = \frac{1}{k*k}\sum_{x=0}^{k*k} ||G_t(j)(x) - G_s(i)(x)||_2^2. \qquad (8)$$

While for orientation control, since now $G_s(i)(x)$, $G_t(j)(x)$ are 2D vectors, we compute the cosine distance:

$$d_{ij}^{GC} = \frac{1}{k*k}\sum_{x=0}^{k*k}\left(1 - \frac{|G_t(j)(x)\cdot G_s(i)(x)|}{|G_t(j)(x)|\cdot|G_s(i)(x)|}\right). \qquad (9)$$

For separate control, we set λ_{GC} as 10 for progression, and 5 for orientation. For achieving two controls simultaneously, we sum up the two distances above and set their weights as 10 and 1. Since there's no neural methods developed for these two controls, we compare our method to [39].

We used the implementation from Neural Doodles [6].

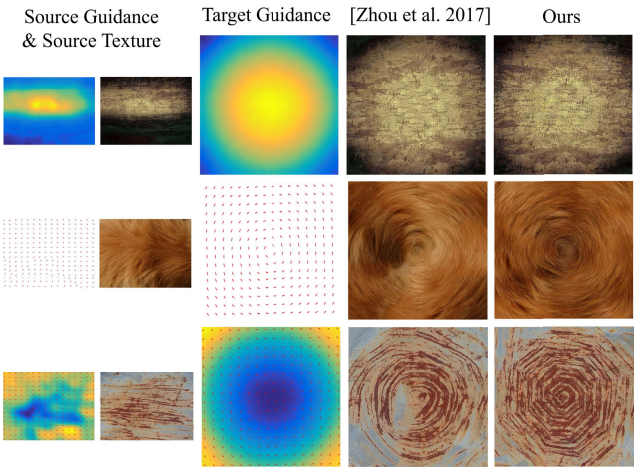

Figure 8. We can realize controls of progression (top), orientation (middle), and two simultaneously (bottom) for inhomogeneous textures as [39] does. Due to the limit of patch augmentation, our results are not smooth enough for the continuous changes in orientation control. But note that [39] needs almost 90 minutes in two controls, while our method requires only 20 minutes.

From Figure 8, we can see our results are completely comparable to those of [39]. One may notice that the orientation changes in our results are not as smooth as theirs. This is because we only use eight orientation augmentations for each source patch as depicted in Sec. 3.3. In contrast, [39] can search the orientation space in any resolution via generalized PatchMatch [2], but needs long-time optimization.

4.4. Real-time synthesis

Beyond texture optimization, the proposed Guided Correspondence loss can serve as a general textural loss in training generative models. Once trained, new textures can be synthesized in real-time by a single forward pass. Following [16], we first tried to train TextureNets [34] on a given stationary texture by replacing the Gram loss with our new loss function. We keep all the settings default as we set for texture optimization, except the multi-resolution strategy. Figure 9 shows a few examples. We can see our loss performs comparable with the Sliced Wasserstein loss.

We then tried a more challenging task that uses the proposed loss to train generative models for real-time controlled synthesis. We take SPADE [27] as the backbone. Conditions such as progression maps or orientation fields are gradually fed into SPADE generator to modulate feature activations for output control. Take progression control as an example. The challenge here is that the network must have a strong generalization ability, given that the user-specified target is probably very different from the source progression. To address that, we separate the training into two stages. In the first stage, we train the model

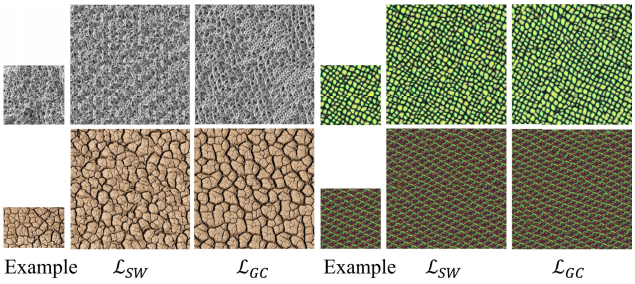

Figure 9. Given an example texture, we train TextureNets [34] either using the Sliced Wasserstein loss ($\mathcal{L}_{SW}$) or the proposed Guided Correspondence loss ($\mathcal{L}_{GC}$), respectively, to update the weights. The network learns to generate textures of size 256×256 during training and produces 512×512p images for inference. Visually comparable results are produced by the two losses.

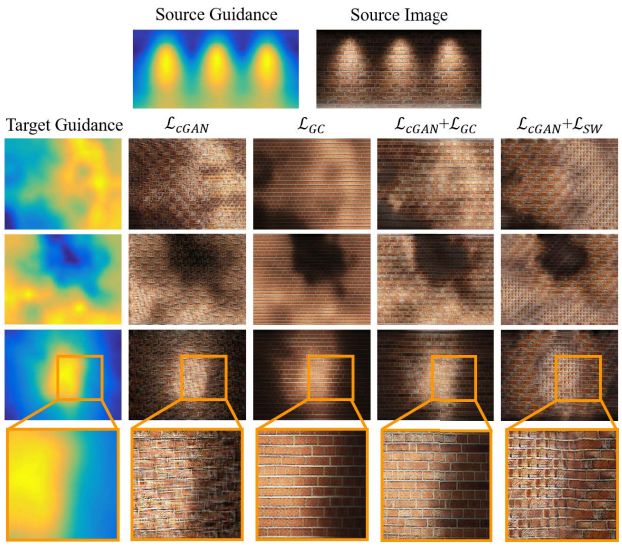

Figure 10. Utilizing the Guided Correspondence loss to train SPADE [27], a conditional generative model, for real-time controlled synthesis. We can see $\mathcal{L}_{cGAN}$+$\mathcal{L}_{GC}$ performs best for progression control, while the other options lead to degraded results.

to learn how to reconstruct the source textures locally from cropped source progression maps. In the second stage, we randomly generate a massive number of target progressions as data augmentation. Since now the synthesized image has no "ground truth", we complement the Guided Correspondence loss ($\mathcal{L}_{GC}$) to the conditional-GAN loss ($\mathcal{L}_{cGAN}$). As shown in Figure 10, without $\mathcal{L}_{GC}$, the conditional GAN cannot generate correct texture patterns. If we remove the discriminator (i.e., use $\mathcal{L}_{GC}$ only), the model suffers from degeneration of fine details. The combination of $\mathcal{L}_{cGAN}$ and $\mathcal{L}_{GC}$ achieves the best result. We also tried to replace $\mathcal{L}_{GC}$ with the Sliced Wasserstein loss ($\mathcal{L}_{SW}$) in the second stage but got degraded results. Figure 1 shows two more examples of real-time synthesis, respectively, on pro-

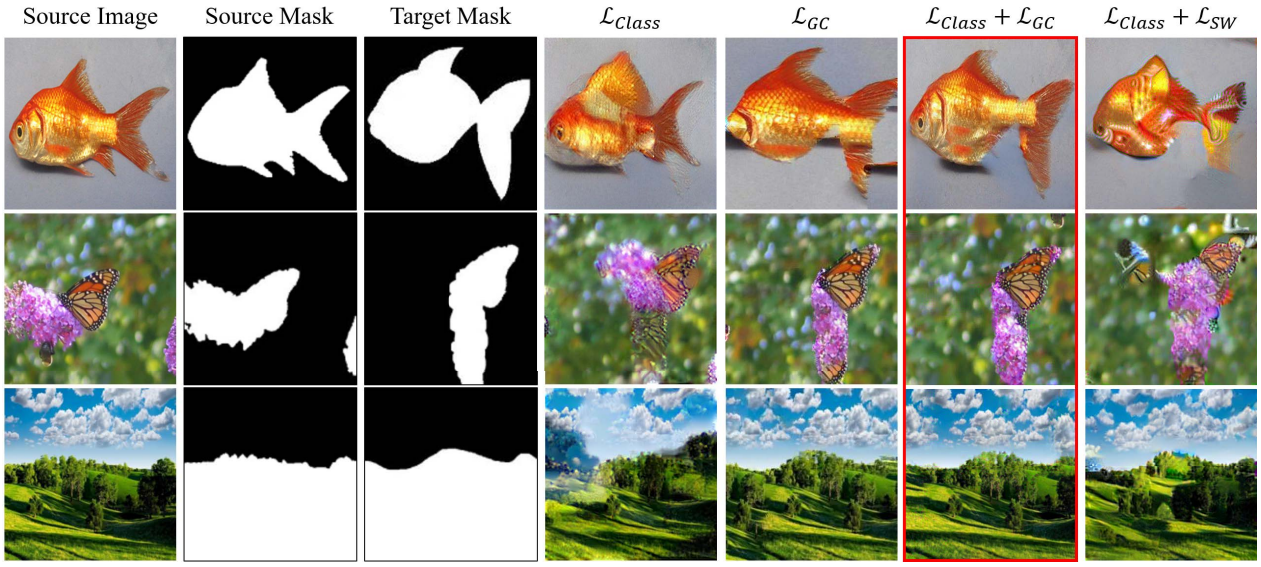

Figure 11. Single-image editing based on inversion synthesis of [36] . $\mathcal{L}_{Class}$ stands for the inversion using classification error. As shown, it cannot guarantee plausible local textures. Using the Guided Correspondence loss only ($\mathcal{L}_{GC}$) produces high-quality textures but with semantic errors, such as the incomplete fish and the butterfly wing inside the flower. The combination $\mathcal{L}_{Class}$+$\mathcal{L}_{GC}$ leads to the most plausible results both in semantic content and fine textures. Using the Sliced Wasserstein loss, however, cannot achieve the same effect.

gression and orientation control, which are trained with the same settings. For more details about how we train SPADE and more results, please refer to the supplementary.

4.5. Image editing

As shown in Figure 12, our method can be easily applied for texture transfer and image inpainting. For the former one, we replace the Gram loss in [13] with our Guided Correspondence loss, and balance it with the content loss by 1:10. For image inpainting, we just need to constrain the texture optimization to fill the holes only using source patches from the remaining area of the same image.

A more interesting while challenging task is single-image editing. Recent single-image models [10, 14, 30] can be applied for editing tasks by maintaining the patch distribution. However, the semantic correctness of output is guaranteed. Recently, Wang *et al*. [36] proposed IMAGINE, an inversion model that uses classification error from a pre-trained ResNet [15] to regularize the results' semantic info. But it cannot produce high-quality textures. To address that, we add the Guided Correspondence loss to the inversion. As shown in Figure 11, the combination of $\mathcal{L}_{Class}$ and $\mathcal{L}_{GC}$ (with weights 1:10) leads to the most plausible editing both in semantic content and fine textures.

5. Conclusion

We have improved the CNNMRF model with a *Guided Correspondence Distance* that takes the matching diversity and additional guidance channels into account, and

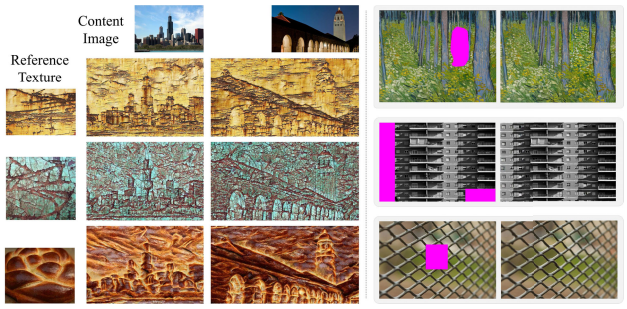

Figure 12. Our method can be easily extended for texture transfer (left) and image inpainting (right).

a *Guided Correspondence Loss* that better optimizes output patches. Intensive experiments with high-quality texture synthesis show that our method is comparable to traditional MRF texture optimization and surpasses existing neural methods, especially in controlled scenarios. The versatility of the proposed loss is demonstrated in training generative networks and single-image editing.

Acknowledgements This work was supported in parts by NSFC (U21B2023, U2001206, 62161146005), NSF of Guangdong Province (2022A1515010221), DEGP Innovation Team (2022KCXTD025), Shenzhen Science and Technology Program (KQTD20210811090044003, RCJC20200714114435012), and Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ).

References

[1] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. on Graphics (Proc. of SIGGRAPH)*, 28(3), jul 2009. 1, 2

[2] Connelly Barnes, Eli Shechtman, Dan B Goldman, and Adam Finkelstein. The generalized PatchMatch correspondence algorithm. In *Proc. of Euro. Conf. on Computer Vision*, 2010. 4, 7

[3] Rachele Bellini, Yanir Kleiman, and Daniel Cohen-Or. Time-varying weathering in texture space. *ACM Trans. on Graphics (Proc. of SIGGRAPH)*, 35(4):1–11, 2016. 2

[4] Saguy Benaim, Ron Mokady, Amit Bermano, and Lior Wolf. Structural analogy from a single image pair. In *Computer Graphics Forum*, volume 40, pages 249–265. Wiley Online Library, 2021. 2

[5] Urs Bergmann, Nikolay Jetchev, and Roland Vollgraf. Learning texture manifolds with the periodic spatial gan. In *Proc. of IEEE Int. Conf. on Machine Learning*, page 469–477, 2017. 1, 2

[6] Alex J. Champandard. Semantic style transfer and turning two-bit doodles into fine artworks. *CoRR*, abs/1603.01768, 2016. 1, 2, 6

[7] Soheil Darabi, Eli Shechtman, Connelly Barnes, Dan B. Goldman, and Pradeep Sen. Image melding: Combining inconsistent images using patch-based synthesis. *ACM Trans. on Graphics (Proc. of SIGGRAPH)*, 31(4), jul 2012. 1, 2

[8] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proc. of SIGGRAPH*, pages 341–346, 2001. 2

[9] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In *Proc. of Int. Conf. on Computer Vision*, 1999. 2

[10] Ariel Elnekave and Yair Weiss. Generating natural images with direct patch distributions matching. In *Proc. of Euro. Conf. on Computer Vision*, volume 13677, pages 544–560. Springer, 2022. 8

[11] Anna Frühstück, Ibraheem Alhashim, and Peter Wonka. Tilegan: synthesis of large-scale non-homogeneous textures. *ACM Trans. on Graphics (Proc. of SIGGRAPH)*, 38(4):1–11, 2019. 2

[12] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2015. 1, 2

[13] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015. 8

[14] Niv Granot, Assaf Shocher, Ben Feinstein, Shai Bagon, and Michal Irani. Drop the GAN: in defense of patches nearest neighbors as single image generative models. *CoRR*, abs/2103.15545, 2021. 8

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of IEEE Conf. on Computer Vision & Pattern Recognition*, pages 770–778. IEEE Computer Society, 2016. 8

[16] Eric Heitz, Kenneth Vanhoey, Thomas Chambon, and Laurent Belcour. A sliced wasserstein loss for neural texture synthesis. In *Proc. of IEEE Conf. on Computer Vision & Pattern Recognition*, June 2021. 1, 2, 5, 6, 7

[17] Nikolay Jetchev, Urs Bergmann, and Roland Vollgraf. Texture synthesis with spatial generative adversarial networks. *arXiv preprint arXiv:1611.08207*, 2016. 2

[18] Alexandre Kaspar, Boris Neubert, Dani Lischinski, Mark Pauly, and Johannes Kopf. Self tuning texture optimization. *Computer Graphics Forum (Proc. of Eurographics)*, 34(2):349–359, may 2015. 1, 2, 3, 5

[19] Johannes Kopf, Chi-Wing Fu, Daniel Cohen-Or, Oliver Deussen, Dani Lischinski, and Tien-Tsin Wong. Solid texture synthesis from 2d exemplars. *ACM Trans. on Graphics (Proc. of SIGGRAPH)*, 26(3):2–es, jul 2007. 1

[20] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. *ACM Trans. on Graphics (Proc. of SIGGRAPH)*, August 2005. 1, 2

[21] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM Trans. on Graphics (Proc. of SIGGRAPH)*, 22(3):277–286, jul 2003. 2

[22] Chuan Li and Michael Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *Proc. of IEEE Conf. on Computer Vision & Pattern Recognition*, June 2016. 1, 4, 5

[23] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *Proc. of Euro. Conf. on Computer Vision*, pages 702–716, 2016. 2

[24] Yitzchak David Lockerman, Basile Sauvage, Rémi Allègre, Jean-Michel Dischler, Julie Dorsey, and Holly Rushmeier. Multi-scale label-map extraction for texture synthesis. *ACM Trans. on Graphics (Proc. of SIGGRAPH)*, 35(4):140:1–140:12, July 2016. 2

[25] M. Lukáč, J. Fišer, P. Asente, J. Lu, E. Shechtman, and D. Sýkora. Brushables: Example-based edge-aware directional texture painting. *Computer Graphics Forum (Proc. of Pacific Conf. on Computer Graphics & Applications)*, 34(7):257–267, oct 2015. 1, 2

[26] Roey Mechrez, Itamar Talmi, and Lihi Zelnik-Manor. The contextual loss for image transformation with non-aligned data. In *Proc. of Euro. Conf. on Computer Vision*, 2018. 2, 3, 4

[27] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proc. of IEEE Conf. on Computer Vision & Pattern Recognition*, 2019. 7

[28] Carlos Rodriguez-Pardo and Elena Garces. Seamlessgan: Self-supervised synthesis of tileable texture maps. *IEEE Trans. Visualization & Computer Graphics*, 2022. 2

[29] Amir Rosenberger, Daniel Cohen-Or, and Dani Lischinski. Layered shape synthesis: Automatic generation of control maps for non-stationary textures. *ACM Trans. on Graphics (Proc. of SIGGRAPH)*, 28(5):107:1–9, Dec. 2009. 2

[30] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *Proc. of Int. Conf. on Computer Vision*, 2019. 1, 2, 5, 8

[31] Omry Sendik and Daniel Cohen-Or. Deep correlations for texture synthesis. *ACM Trans. on Graphics*, 36(5):161:1–161:15, July 2017. 2

[32] Wu Shi and Yu Qiao. Fast texture synthesis via pseudo optimizer. In *Proc. of IEEE Conf. on Computer Vision & Pattern Recognition*, pages 5498–5507, 2020. 2

[33] Assaf Shocher, Shai Bagon, Phillip Isola, and Michal Irani. Ingan: Capturing and retargeting the "dna" of a natural image. In *Proc. of Int. Conf. on Computer Vision*, 2019. 1, 2

[34] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *Proc. of IEEE Int. Conf. on Machine Learning*, page 1349–1357, 2016. 2, 7

[35] Jiaping Wang, Xin Tong, Stephen Lin, Minghao Pan, Chao Wang, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Appearance manifolds for modeling time-variant appearance of materials. *ACM Trans. on Graphics (Proc. of SIGGRAPH)*, 25(3):754–761, 2006. 2

[36] Pei Wang, Yijun Li, Krishna Kumar Singh, Jingwan Lu, and Nuno Vasconcelos. IMAGINE: image synthesis by image-guided model inversion. In *Proc. of IEEE Conf. on Computer Vision & Pattern Recognition*, pages 3681–3690. Computer Vision Foundation / IEEE, 2021. 8

[37] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proc. of SIGGRAPH*, pages 479–488, 2000. 2

[38] Yonatan Wexler, Eli Shechtman, and Michal Irani. Space-time completion of video. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 29(3):463–476, 2007. 2

[39] Yang Zhou, Huajie Shi, Dani Lischinski, Minglun Gong, Johannes Kopf, and Hui Huang. Analysis and controlled synthesis of inhomogeneous textures. *Computer Graphics Forum (Proc. of Eurographics)*, 36(2):199–212, 2017. 1, 2, 6, 7

[40] Yang Zhou, Zhen Zhu, Xiang Bai, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Non-stationary texture synthesis by adversarial expansion. *ACM Trans. on Graphics (Proc. of SIGGRAPH)*, 37(4):49:1–49:13, 2018. 1, 2, 4, 5