Contents lists available at ScienceDirect

Computers & Graphics (Pre-print)

journal homepage: www.elsevier.com/locate/cag

Automatic Extraction and Synthesis of Regular Repeatable Patterns

Carlos Rodriguez-Pardo[1,1,*], Sergio Suja[1,1], David Pascual[1,1], Jorge Lopez-Moreno[1,1], Elena Garces[1,2,*]

ARTICLE INFO

Article history:
Received July 11, 2019

Keywords: Computers and Graphics, Computer vision, Image representations, Interest point and salient region detection, Shape inference, Machine Learning

ABSTRACT

Textures made of regular repeating patterns are ubiquitous in the real world, most notably in man-made environments. They are defined by the presence of a repeating element, which can show a significant amount of random variations, non-rigid deformations or color noise. We propose an end-to-end pipeline capable of finding the size of the minimal repeating pattern in single images, as well as obtaining the single repetition that, when tiled, produces the most similar synthesis to the complete image. We do this by combining state-of-the-art algorithms in image transformations, repeating pattern detection, image stitching and deep perceptual losses. Additionally, we show how our pipeline can find the minimal color pattern in woven fabrics, which can be useful for both surface-based render methods and computer vision tasks in the textile domain.

1. Introduction

Texture synthesis is one of the classical problems in the computer graphics field, and numerous techniques exist which are able to successfully replicate textures with different degrees of regularity. Nowadays, these techniques have become particularly relevant for efficiently generating content for Virtual Reality environments. In particular, texture mapping is a technique widely used in real-time rendering engines to improve the realism of materials by warping textures into the 3D models. In order to do this operation effectively—with low memory consumption— the texture must be *tileable*, that is, it should allow the concatenation of multiple copies of itself without producing visible artifacts at the boundaries.

We propose an end-to-end method to extract the minimum representative tile (or pattern) of an input image, which replicates its original appearance when tiled. Generating these kind of *tileable* textures is a problem that has received very little attention. Only recently, Moritz et al. [1] propose a solution to synthesize non-stationary tileable textures from pictures. Our work is complementary to theirs, that is, we aim to detect the minimum pattern that represents the image, which allow us to efficiently generate a novel image given repetitions of the pattern as well as fit regular lattices. We take as input fronto-planar geometrically-regular textures [2], which makes our algorithm particularly suitable to detect the weaving color structure of fabrics (as shown in Figure 1) or to identify representative elements in facades.

Our method builds on recent works that explore the feature spaces of neural networks to perform image interpolation [3], or lattice detection [4]. We leverage the correspondence between the activation of features in the deep spaces of neural networks and the location of its triggering pixels to find repeating structures. By means of a voting process, we provide the size of the minimum tile and propose the optimal tile that represents the image. To this end, we also apply state of the art perceptual metrics to automatically evaluate the quality of the synthesized image versus the original input.

*Corresponding authors:
e-mail: carlos.rodriguez@seddi.com, elena.garces@urjc.es ()
[1]Seddi Labs (Madrid, Spain)
[2]Universidad Rey Juan Carlos (Madrid, Spain)

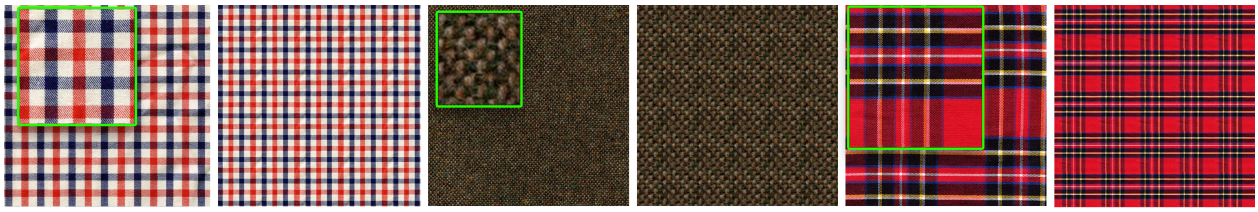

Fig. 1. Examples of three fabrics with the minimum pattern found by our algorithm (inset, in green), and novel syntheses produced by our method. Note that the rendered images are the result of seamless tiling the minimum repeating pattern of the original image.

Our contributions are the following:

- We propose a novel end-to-end pipeline that finds the size of the minimum repeatable pattern of an image.
- Given a candidate pattern size, we propose an algorithm that, using a state of the art deep perceptual loss, jointly finds and synthetize the optimal tileable pattern that better reproduces the input image if concatenating multiple copies of itself.
- We show how our pipeline can be used to extract the minimal color structure in pictures of fabrics, as well as most frequent architectural elements in pictures of facades.

2. Related work

We are unaware of methods that can perform end-to-end automatic extraction and synthesis of repeatable patterns in single images using a deep learning approach, as our method is capable of. Nevertheless, our work is closely related to several topics in computer vision and graphics, including texture synthesis or image blending.

Texture synthesis. Texture synthesis algorithms are typically concerned with creating synthetic images that resemble one or multiple exemplar images that contains different degree of regularities [2]. A seminal work on this topic [5] proposes an optimization-based texture stitching algorithm that performs the synthesis by finding the optimal way of stitching different parts of the input image with copies of itself. More recently, Kaspar et al. [6] proposed a self-tuning algorithm that can perform photo-realistic texture synthesis by automatically weighting *guidance channels*.

Convolutional neural networks have been extensible used in texture synthesis and style transfer. Gatys et al. [7] performs texture synthesis by minimizing a loss function which compares the deep latent space of the exemplar and the synthesized images, using gradient descent. A similar approach exploiting feature correlations was used by Sendil & Cohen-Or [8]. Finally, Zhou et al. [9] propose a generative method which can perform texture synthesis on regular or non-regular exemplars by training a deep generative network to expand small crops of the input image. Our method is loosely based on deep texture synthesis methods. We make use of the feature spaces of neural networks for repetitive pattern extraction and for finding an optimal tile that can represent the greatest amount of information in the input image as possible. We have found that our method is capable of performing texture synthesis of geometrically regular and near-regular textures, but it is not suitable for stochastic textures.

Repeated pattern detection. Finding repeating patterns is a challenging task, in part because there is no clear definition of what a repeating pattern means in the visual domain. A common approach is to assume that the repetitive patterns lie on a 2-dimensional lattice [10], or in a planar structure on the image [11, 12]. In terms of image descriptors used to represent image information such as shapes or colors, we find a vast amount of methods [13, 12, 14], that rely on *Scale-invariant feature transforms* (SIFT) [15] as feature descriptors.

Despite the popularity of SIFT for this problem, the emergence of pre-trained deep convolutional neural networks that contain filters learned using natural images have created new opportunities in this field. In particular, Lettry et al. [4] presents a method to find repetitive patterns by looking for spatial regularities in the activations of filters in a pre-trained neural network. Their method is capable of finding the most probable size of the repeating pattern in the image, as well as fitting a grid that can segment those repeating patterns. A main disadvantage of this method is that, unlike in [12], the repeating pattern can only be of one size, and it works better when the grid in which the repeating pattern lies is aligned with the axis of the image and when the image is fronto-planar. Despite those disadvantages, we find their work to be a suitable starting point that inspired our algorithm for geometrically regular input textures.

Image stitching. The goal of these methods is to combine different images with overlapping parts to create a larger image in a seamless way, i.e., it is not possible to tell where the images were blended. One of the seminal work in the field [5] is based on minimizing differences between overlapping blocks to find the optimal cuts, resulting in a near-seamless stitched image. More recent methods simply blend the borders of the images, but they enforce continuity and smoothness in the gradients at different scales in the parts of the images that were blended [16, 17, 18]. This kind of blending is more useful in situations where the images are very similar to each other. There are sophisticated algorithms that can perform this task, by exploiting symmetry [19], using graph cuts [20] or melding image patches [21]. In our method, we simply want to stitch one image with a copy of itself, so we do not need to deal with the amount of irregularities that those last three methods can handle.

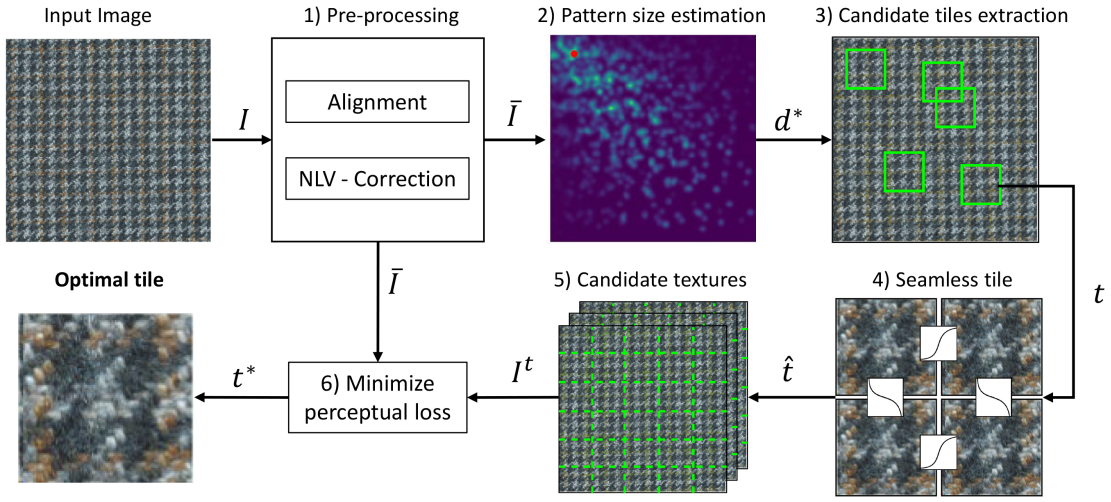

Fig. 2. Illustration of our pipeline. 1) The input image is pre-processed by aligning it with the main axis and removing non-local variations [22]. 2) Using CNN activations, we find the most consistent *displacement vector*, which suggests the size of the repeating pattern in the image. 3) We obtain a list of candidate regions in which the repeating pattern is present. 4) We make each of the candidate tiles tileable with Gaussian blending and template matching. 5) We synthesize one full texture with each candidate tile and 6) compare it with the original input image using a perceptual loss. The region that showed the smallest perceptual error with respect to the input image is the output of our algorithm.

3. Overview

Our goal, as illustrated in Figure 2, is to identify the minimum repeatable structure that —when tiled— can replicate most of the overall appearance of the input image. To this end, we leverage state-of-the-art algorithms and machine learning models for image processing. Initially, as explained in Section 4, we pre-process the input image by aligning it with the main xy-axis and regularize its appearance by removing non-local variations. Then, we use the feature space of a convolutional neural network to find the size of the minimal repeating pattern in the image (Section 5.1) and a list of candidate repetitive tiles (Section 5.1). For each of these candidate tiles, we create a seamlessly tiled texture of the same size as the input image using a combination of template matching and Gaussian blending algorithms (Section 6.1). Finally, we find the tile that better represents the image by comparing the tiled image with the original input image using a state-of-the-art perceptual error metric (Section 6.2). Hereafter, we explain why and how each of those steps is performed.

4. Pre-processing

The input to our method belongs to the class of geometrically regular near-regular textures, i.e. a single texton recurring on a lattice. In particular, we require the image to be fronto-parallel to the camera which makes the method particularly suitable for captures of fabrics and facades. Although the input images are in general homogeneous, they might contain small inconsistencies or irregularities due to wrinkles, dust or noise. We therefore apply an initially pre-processing step to align the image with the main xy-axis and make the sample more homogeneous.

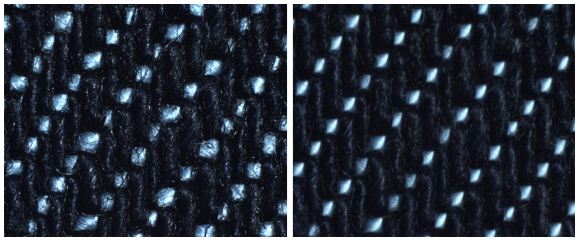

Fig. 3. Example of a *regularized* image. Left: The input image, with irregular yarns and non-homogeneous color variations. Right: The *regularized* image, where those irregularities are hidden by the NLV algorithm [22]. There is a trade-off between how many details can be kept in the output image and how regular it is.

Image Alignment. Aligning the input texture with the main axis is a key initial step as, otherwise, the task of finding those repeating patterns would become significantly more challenging. Our goal is to find the angle α such that when we rotate the input image I by α degrees, the rotated image is axis-aligned. As suggested in previous work [23, 24, 25], we use the Radon Transform of a two-dimensional function for a given angle ϕ: $RT_\phi(x, y)$. We first filter the input image I using a Sobel operator to emphasize edges, obtaining $\mathcal{F}(I)$. Then, for an evenly space set of 1000 angles in the range of $\phi \in [-45, 45)$ degrees, we compute the Radon Transform $RT_\phi(\mathcal{F}(I))$. Finally, α is chosen using the following equation: $\alpha = \arg\max_\phi \frac{d\sigma_\phi}{d\phi}$, where σ_ϕ is the standard deviation of $RT_\phi(\mathcal{F}(I))$.

Image Regularization. Frequently, textures present unwanted irregularities in their structure due to, for example, geometrical variations. Furthermore, it is also possible to find undesirable elements, such as particles of dust or wrinkles, that are introduced in the capture process and hinder the task of finding the optimal repeatable tile. In order to make the search of the optimal tile more robust to this kind of noise, we make use of the

Non-Local Variations (NLV) algorithm from Dekel et al. [22], which iteratively transforms the input image I into one in which the internal recurrence is maximized. To do so, it uses patch-based processing [26] to find similar patches and optical flow to find the smooth transformation that maximizes the presence of those recurrent patches. In our experiments, we ran the algorithm for 200 iterations and we set the hyper-parameters λ and α to 0.1, and γ to 30. Those parameters control how similar the original and the regularized image are to each other. Please refer to the original publication for full details on the algorithm. We show in Figure 3 and Figure 7 some examples of the regularization and its effects in the whole pipeline in the results (Section 7). The result of these two steps of pre-processing is a novel image $\bar{I}$ which is more regular and aligned with the XY axis.

5. Automatic extraction of representative tiles

The goal of this step is to find a set of candidate representative tiles. We are inspired by the work of Lettry et al. [4], which introduces the idea that when a pattern is regularly repeated in a grid, the same filters in a deep convolutional neural network activate in regularly-spaced regions of the image. We build on that work and simplify their algorithm to look for the size of the most consistent repeating pattern in an image. While their goal is fitting a lattice using the repeating pattern found by their method, ours is to find the size of the repeating pattern and a list of candidate representative tiles. In Section 5.1, we first present our method to find the size d^* of the minimum tile and in Section 5.2 we show how we identify the list of candidates tiles $\{t\}$.

5.1. Pattern size estimation

Convolutional neural networks learn hierarchical representations of features that are useful for many tasks, including image classification, object detection or style transfer [27, 28, 29]. This representation of features is done in the form of convolutional filters, that can be used to look for specific patterns in images. Moreover, deep CNNs learn features that increase in complexity as deeper layers depend on the filters learned by previous layers in the network [30]. For example, the first layers of a deep CNN usually learn to detect edges and other low level features, whereas deep layers tend to learn higher-level representations of features, such as objects or complex shapes. Consequently, a deep CNN can be considered as a set of filters with different levels of abstraction that activate when a particular feature is present on a region of an image. A main advantage of using deep CNNs instead of manually engineering a set of filters is that they can learn those filters from data, thus making them more robust and independent from human assumptions or prior knowledge of the problem. Leveraging the multi-level capability of CNNs filters, the key idea of this step is to find repetitions in image space of peaks of activations in filter space. The most frequent distance between peak activations in both image and feature spaces is the most probable size of the minimal tile.

In every layer of a deep CNN, there are filters that activate more than others, because there are features that are more present than others in the input image. To reduce the amount of computation in our algorithm, we only keep the set of filters F_l with meaningful responses as follows:

$$F_l = \{f_l \mid \mu_{f_l} > \delta \cdot \max(l)\} \tag{1}$$

where $f_l \in F_l$ is a filter of a layer $l \in L$, L is the set of convolutional layers in the deep CNN, μ_{f_l} is the mean activation of each filter, $\max(l)$ is the maximum μ_{f_l} in the layer l, and $\delta = 0.65$ [4].

To compute the size of the repeating pattern, we feed the regularized image to our deep CNN. Then, for each f_l, we find the set of pixels P_{f_l} that correspond with activation peaks, i.e. the pixels in the input image that maximally activate f_l: $P_{f_l} = \{p \in I \mid f_l^p \geq 2\sigma_{f_l} + \mu_{f_l}\}$, where f_l^p is the activation value of pixel p in layer l, μ_{f_l} is the mean activation of each filter, and σ_{f_l} its standard deviation. Then, we find a set D_{f_l} of *displacement vectors* by computing a vector for each pair of peak activations in each filter $D_{f_l} = \{d^{i,j} = |p_i - p_j| \mid \forall p^i, p^j \in P_{f_l}, i > j\}$, where $|\cdot|$ denote element-wise absolute value on vectors. Each of the *displacement vectors* $d^{i,j} \in D_{f_l}$ is a candidate tile size.

Given that there are different numbers of filters in each layer in our model, we need to normalize how much each vector in each filter will contribute to the final estimation of the optimal tile size. Thus, we divide each vector by the number of vectors $|D_{f_l}|$ in each layer and filter. Each displacement vector contributes to a Hough voting space $\mathcal{V} : \mathbb{R}^2 \rightarrow \mathbb{R}$, using the formula:

$$\mathcal{V}_{[x,y]} = \sum_{l \in L} \sum_{f_l \in F_l} \frac{1}{|D_{f_l}|} \sum_{d^{i,j} \in D_{f_l}} d^{i,j} \tag{2}$$

$\mathcal{V}$ is smoothed using a 2-dimensional Gaussian filter [31] with a kernel size of $(13, 13)$. The optimal displacement vector, which has the dimensions of our candidate minimal tile, is defined as:

$$d^* = (\arg\max_x \mathcal{V}_{[x,0]}, \arg\max_y \mathcal{V}_{[0,y]}) \tag{3}$$

We restrict the size of d^* so $d_a^* > \frac{dim_a}{20}, a \in \{x, y\}$, where dim_a is the dimension in pixels of the input image in the axis a. We illustrate this voting space in Figure 4. Given that we are using images with a size of 227×227, we force the displacement vector to be of at least 12 pixels in each dimension.

In comparison with Lettry's model, we simplify their displacement vector model significantly, as we do not assume that each vector follows a bi-variate Gaussian distribution. Consequently, we do not need to assume that there are different standard deviations for each layer in the network, which reduces the number of hyper-parameters that need to be tuned. Moreover, we do not take into account filters that are not significantly activated (Equation 1). We discard those filters for the whole tile extraction process, whereas they do not consider them in later stages of their pipeline, when building their *implicit pattern model*.

5.2. Candidate tiles extraction

Using the optimal tile size d^* found by the previous step of the algorithm, we now want to find the regions of the input image that are more consistent with it. That is, we want to find the pixels in the input image that correspond to the activation peaks that are most consistent with the optimal displacement vector. This process will help us find candidate crops of the input image that, when tiled, can replicate the whole appearance of the input image.

For all the filters $f_l \in F_l$, we exponentially weight each filter according to how consistent their displacement vectors $d^{i,j} \in D_{f_l}$ are with the optimal vector d^*. The weights are also normalized using the number of vectors in each filter $|D_{f_l}|$ as follows:

$$w_{f_l} = \sum_{d^{i,j} \in D_{f_l}} \left(\frac{1}{|D_{f_l}| + \lambda} \right) \exp\left(-\frac{\|d^{i,j} - d^*\|^2}{2\sigma^2} \right) \tag{4}$$

where the parameter λ is a prior on the number of expected repetitions, and is set to 0.8 following previous work [4]. On the contrary, as opposed to previous work that require a different σ per layer, we set σ to 1 for every layer so as to avoid making some layers more important than others. We empirically found that this value of σ yielded results comparable to those of previous work, without the need of optimizing the value of this parameter. Besides, our data showed regular patterns of different levels of abstraction, which are typically represented by different layers in the neural network. To avoid biasing one level of abstraction over others, we set the same value to all the layers.

Once we know, for each layer, how consistent their displacement vectors are with respect to the optimal vector, we recompute the points in which each of the filters were maximally activated, and, for each pixel p in the input image, we compute the weighted sum of all the activation peaks $p \in P_{f_l}$, obtaining a weighted map $M(x, y)$ of activations (see Figure 4):

$$M(x, y) = \sum_{l \in L} \sum_{f_l \in F_l} (w_{f_l} \cdot p(x, y)) \tag{5}$$

This map M can be understood as a map of probabilities that shows how likely each pixel (x, y) is to be the point in which a candidate tile starts. Using a standard local maxima extraction algorithm, (using a neighborhood side of 30), we obtain a small list of candidate points C. Note that both the candidate points and the size of the displacement vector are represented in the dimensions of the re-scaled image that feeds the network. Consequently, we apply a proportional re-scaling operation to match the size of the original input image. After this operation, we obtain the list of candidate tiles $t_i \in T$ by taking each origin point c_i and extending a window of size $2d^*$:

$$t_i \leftarrow \bar{I}\left[c_i(x) : c_i(x) + 2d_x^*, c_i(y) : c_i(y) + 2d_y^* \right] \forall c_i \in C \tag{6}$$

This step is different from previous work, which fit an *Implicit Pattern Model*. This model groups points together by computing a centroid of points so as to reduce the number of extracted tiles.

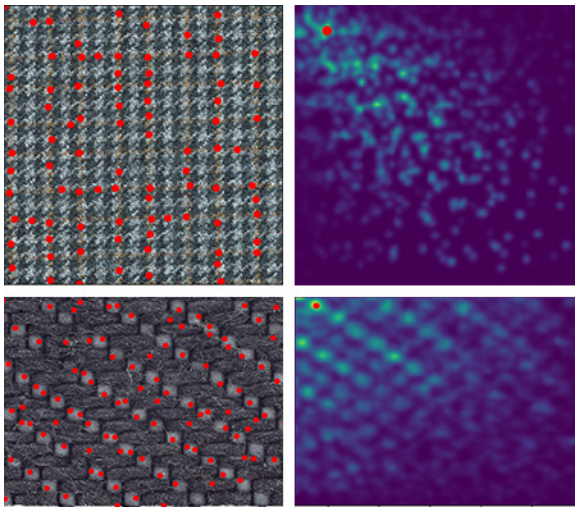

Fig. 4. Illustration of the displacement vector and activation map extraction. Left: Original input image where the local maxima of the activation map are represented using red dots. Right: Computed displacement vector map. Yellower points represent more consistent votes than blue points. The red dot is the optimal d^*. It is worth noting how the red dots tend to follow a grid that matches the yarns in the fabric, which suggests that the network has been maximally activated by yarns.

6. Optimal tile extraction and synthesis

Once the set T of candidate tiles has been obtained, the goal is to find the optimal tile $t^* \in T$ such that, when tiled, it looks as close as possible to the input image. This is challenging as not all tiles are equally suitable for tiling. That is, although they have the same size, they represent a different region of the image which might contain different amount of noise and non-regular variations. The straightforward process of stitching one image to itself might yield inconsistencies in the transition gradients. This can be solved as an optimization problem that guarantees smooth gradients and apply warping techniques to maintain structural regularity [1, 32], however, these algorithms tend to be computationally very complex and inefficient. Our method for finding the optimal tile aims to solve these two problems in a simple and efficient way: We tile the image using Gaussian blending and template matching (Sections 6.1) until it fits the size of the original image, and then compare each tiled image with the input image with a perceptual loss (Section 6.2).

6.1. Gaussian blending for seamless tiling

In this section, we propose a method which is capable of transforming our t_i into seamlessly tileable structures, with a combination of template matching and Gaussian blending. Let t_l and t_r be two copies of the same tile, that we want to blend together horizontally at an optimal position with a smooth transition, obtaining t_b.

First, we extend the borders of the tile by a 10% in each dimension by using the original image, we call this the *extended boundary*. Both t_l and t_r become $(n \times m)$ dimensional images. Then, using cross-correlation template matching [33], we look for the optimal position of the overlapping tiles restricting the search to the extended boundary, and also restricting the direction of the displacement in the X and Y axis for horizontal and

vertical boundaries, respectively. As a result of this step, we obtain a new image which results from stitching the two tiles at the optimal position. Assuming the overlapped area has $(d_x \times m)$ dimensions, our blended image t_b has a size of $(2n - d_x, m)$ pixels, and is defined as follows:

$$t_b[0 : n - \frac{d_x}{2}; 0 : m] \leftarrow t_l[0 : n - d_x; 0 : m] \tag{7}$$

$$t_b[n : 2n - \frac{d_x}{2}; 0 : m] \leftarrow t_l[d_x : n; 0 : m] \tag{8}$$

In the overlapped area we alpha-blend [34] both images with a variable alpha that depends on the distance to the boundary following a univariate Gaussian distribution, which is symmetrical around its mean:

$$t_b((n - d_x) + x, y) \leftarrow \alpha(x)t_r(x, y) + (1 - \alpha(x))t_l((n - d_x) + x, y); \tag{9}$$

$$\alpha(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx \tag{10}$$

where $0 \leq x \leq d_x$, $0 \leq y \leq m$, $\mu = \frac{d_x}{2}$ and $\sigma = 2$. The value of σ controls how smooth the transformation is. The smaller the value of σ, the less seamless the tiling will be, but more detail is preserved. An equivalent method is used to blend the images vertically, resulting in an overlap of size $(2n - d_x, d_y)$ and a t_b of total size $(2n - d_x, 2m - d_y)$. As t_b has four repetitions of the original t_i, we transform t_i by cropping t_b so we only take into account one repetition:

$$t_i \leftarrow t_b[\frac{2n - d_x}{2} : 2n - d_x, \frac{2m - d_y}{2} : 2m - d_y] \tag{11}$$

6.2. Deep perceptual loss

Given the list of generated candidate tiles, we want to choose the best representative of the perceptual information contained in the regularized input image $\bar{I}$. To this end, we define an error metric that summarizes into one scalar number, the difference between a pair of images. Pixel-wise comparisons, such as the L_1 or L_2 norms have been proven to be ineffective for this task [35]. Instead, it has been recently shown that deep perceptual metrics, which compute the difference between images by taking into account how different are the activations in the convolutional layers of a deep CNN, resemble how human perception behaves, better than any other known metrics [35]. They have been successfully used in several computer vision problems, most notably texture transfer or image-to-image translation [36].

We follow previous work [7, 9] and compute a *Style Loss* using the activations of the ReLU layers of a pre-trained *VGG19* deep CNN. As in Zhou et al. [9], we choose the layers *relu*2_1, *relu*3_1, *relu*4_1, and *relu*5_1 as our feature maps. Let $\bar{I}$ be our regularized input image and I^t be the image that is created when tiling the tile t until it has the same dimensions as $\bar{I}$. A layer l in the *VGG* network has N_l filters, each of size M_l. Each feature map can be represented as a matrix F^l, where F^l_{jk} is the activation of the filter j at position k. A Gram matrix of a layer l is defined as the inner product between two feature maps i and j: $G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}$. The *style loss* between two images in a particular layer is defined as:

$$E_l(\bar{I}, I^t) = \frac{\sum_{i,j}(G^l_{ij}(\bar{I}) - G^l_{ij}(I^t))^2}{(2N_l M_l)^2} \tag{12}$$

The total style loss is the weighted sum of the error in all the layers that are considered: $\mathcal{L}_{style}(\bar{I}, I^t) = \sum_l w_l E_l$, as illustrated in figure 5. We weight each of the *ReLU* layers mentioned above using the same values as in [9]: $w_l = \frac{1000}{N_l^2}$.

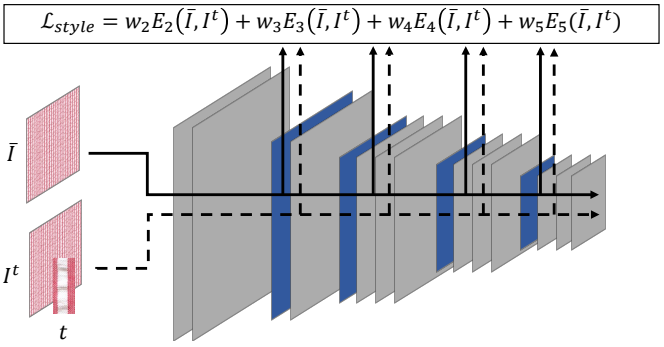

Fig. 5. Illustration of the perceptual loss used in our pipeline. The image I^t is the crop t tiled so that it fits the size of the input image $\bar{I}$. Both images are passed through a VGG-19 network and the latent spaces for the *ReLU* layers are compared using the equations above.

The VGG19 was trained on (224×224) dimensional images. As we did when computing the activation peaks in for the displacement vector calculation, we re-size $\bar{I}$ and I_t to the size that the network expects to receive. We perform this operation adding an adaptive average pooling [37] layer on top of the network, which adapts images of any size to the wanted dimensions. It is worth mentioning that other deep perceptual losses have been proposed in the literature, like a *feature reconstruction loss* [38], but we choose the style loss as it can represent color differences more robustly than losses that are designed to represent differences in structure or shapes. As our input images contain fabrics, which typically show more variation in colors than in structure, the style loss is more suitable for this task.

Optimal tile. We obtain the optimal tile t^* as the one that minimizes the perceptual error between the regularized image $\bar{I}$ and the tiled image I^t:

$$t^* = \arg\min_t \mathcal{L}_{style}(\bar{I}, I^t) \tag{13}$$

7. Results

Implementation. We use an AlexNet [39] pre-trained on the ImageNet [40] dataset as our deep CNN model to compute the repeating pattern size, using the PyTorch framework [41]. This model provides a rich variety of filters and is lightweight compared to deeper models, such as VGG-16 or Inception-V3 [42].

Image ID	Pre-processing	Minimum Tile Estimation and Synthesis				Total
	NLV (Sec 4)	Size (Sec 5.1)	Candidates (Sec 5.2)	Optimal (Sec 6)	Total	
#1	2432	48	1	17	67	2499
#2	2183	57	1	11	70	2253
#3	2059	35	0	9	44	2103
#4	2104	32	1	19	52	2156
#5	1910	16	2	15	34	1944
#6	2308	54	1	12	67	2375
#7	2119	41	2	16	59	2178
Mean	**2159**	**41**	**1**	**14**	**56**	**2215**

Table 1. Time in seconds of every step of our pipeline. The most time consuming part is the pre-processing step due to the NLV regularization algorithm which takes over one hour for images of size 1920x1080. The minimum tile estimation and synthesis takes less than a minute on average. Note that to use NLV regularization is optional and our method can also work without it reasonably well. Please refer to figures 3 and 7 for examples of the influence of using NLV, and Figure 8 for an example without it.

As we need to find where in the input image each filter is maximally activated, using deeper models would increase significantly the time that it takes to find the most consistent repeating pattern, without necessarily increasing the accuracy of our results. This network was trained on 227×227 dimensional images, therefore we re-size our input image to that size and feed it to the network after filtering it with bilateral filter [43] to remove noise but maintain the relative intensity of the gradients in the image. It is worth noting that we are not using the fully connected part of the AlexNet model, which allows us to use images of any size. Nevertheless, the network has only been trained on small images, and, consequently, the filters that it has learned are related to features found in small images and may not translate to images of a greater size. The main bottleneck of our algorithm, as shown in Table 1 is the NLV regularization step (Section 4), which usually takes around 40 minutes for images of size 1920x1080. This is a reasonable resolution for this step since a smaller image would result in an over-smoothed regularized image. We believe its implementation could be optimized but this was not tested. Moreover, if the input image is sufficiently regular, as shown in Figures and 7 and 8, this step is not necessary. Extracting the CNNs activations and the optimal tile usually takes less a minute. In comparison with the original implementation of Lettry's, our algorithm is considerably faster. They do not report exact timings of their experiments, but they mention it takes from *tens of seconds to minutes*, whereas our implementation never surpasses the 60 seconds of computational time.

Qualitative evaluation. Figures 1 and 7 show the results of our algorithm when applied to scanned fabrics. In the majority of the cases, when tiling the optimal tile t^*, the resulting synthesized image can fully replicate the color structure of the whole fabric with high accuracy. By regularizing the image, we observe what was expected: an increase in the structural and color regularity of the image helps the overall performance of our algorithm. This can be seen in the third row of Figure 7: Small shadows and wrinkles in the capture cause a too small size of the tile. In some fabrics, this regularization has the cost of eliminating some information in the input image, but this never translated into a wrong estimation of t^*. The result in the red-striped fabric was interesting. It has structural wrinkles that are detected by our algorithm, but there is no regularity in them. When finding the optimal tile t^*, our perceptual loss function detected that the most optimal tile had three evenly-space wrinkles. This is not the obvious choice, as there are many other candidate tiles with the same size, but, when tiled, the image looks like a realistic fabric. We additionally show in Figure 8 and in the supplementary material the results of our algorithm when applied to mosaics with satisfactory results.

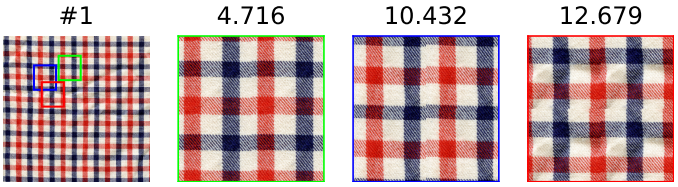

Fig. 6. Example of the result of our algorithm for image #1 in Table 1. From left to right, the tiles with the best, median and worst perceptual error are shown, with their respective $\mathcal{L}_{style}$ values. We show two repetitions of each tile to help the visualization of the artifacts created when tiling.

In terms of the result of the tiling, we can see that, despite the borders of t^* being distorted by our Gaussian blending algorithm, the tiling does not create any unwanted artifacts. Due to the regularity in the data capturing process and its front-parallel nature, the repeating pattern in the images are generally axis-aligned, thus making them ideal for tiling.

When our algorithm is not capable of finding the minimal structure, it is because not enough repetitions were present in the input image, or because the pattern has suffered excessive geometrical deformations. An example of such deformation is shown in Figure 9. In this case the minimal tile is bigger than the real one because of the shadows produced by the wrinkles and the deformation of the pattern. Nevertheless, our algorithm has obtained a tile that creates plausible seamless images with consistent deformations. Finally, in some cases, the extracted optimal tile contains several repetitions of the color pattern. This is a consequence of our constraints on the minimum size of d^*. However, even if the tile t^* is bigger than the actual color pattern, it is always smaller than the original input image, which is an advantage for applications such as rendering, where there are memory requirements that make smaller textures quite desirable.

Quantitative evaluation. In the supplementary material, we report a quantitative evaluation on the quality of the synthesis in terms of its perceptual loss. For the images in Table 1, we show the tiles that yielded the best, worst and median perceptual error $\mathcal{L}_{style}$. This perceptual metric cannot be used to compare the quality of the synthesis of different input images, because the scale of the loss function depends on the size of the tile. Nevertheless, it succeeds on distinguishing between crops of the same image that are easily tileable and those that are harder to tile smoothly. More precisely, it is able to find the tile that replicates the most information in the input image as possible and that is the most seamlessly tileable. For instance, in Figure 6, we can see that the optimal texture can be tiled more smoothly than the other two textures that are reported. We show the results on

the non-regularized images so as to make the differences more visible.

Comparison with Lettry et al. [4] As explained before in Sections 5.1 and 5.2, our pipeline builds upon the work of Lettry et al. However, there are meaningful differences in terms of efficiency and target application. Our goal is to find one representative tile that allow us to seamlessly synthesize new textures. On the contrary, their purpose is to fit a regular lattice by means of identifying points in the image that correspond to the most repetitive elements. We show in Figures 10 and 11 comparisons with such method. The two examples of Figure 10 show similar results for both methods. We build the lattice by extending vertically and horizontally the size of the optimal tile, starting from that tile as the origin. Lettry's method uses RANSAC algorithm to fit an elastic model using the identified maxima. In addition, in Figure 11 we show an example where our simplification fails to identify the most repetitive element. The reason is because their probabilistic model is more robust and able to identify the repeating pattern even in the presence of low frequency details. Our simplification assumes that all layers have the same standard deviation, and thus, the method is fine-tuned for high frequency textures, ignoring patterns of low frequency activations.

8. Conclusions and future work

We have presented an end-to-end pipeline capable of the extraction and synthesis of repeating patterns in single images. Our method combines state-of-the-art image regularization, repeating pattern detection and deep-learning based perceptual losses, making it robust to non-local variations in the structure of the input image, as well as capable of handling repeating patterns of different sizes and shapes. Our results demonstrate that this pipeline is capable of successfully extracting the color pattern in woven fabrics, as well as being able of synthesizing them by adapting image stitching algorithms to this domain. All the methods in the pipeline are domain-agnostic, and, consequently, it should work for any domain in which fronto-planar images with repeating patterns are available.

We have seen that the weighted combination of features in the deep neural network can be used successfully to find color structure in pictures of regular textures. In convolutional neural networks, deeper layers of the network have a bigger receptive field than filters in the early layers of the network and as such, they can detect objects and shapes that are larger than the receptive field of the kernels in the first layers in the network. Consequently, our algorithm has shown to be invariant to the repeating pattern sizes.

Our method could be extended in several ways. First, a future potential application that could benefit from of our technique, as shown in Figure 10, is to find representative elements in pictures of facades [44, 45]. Second, the pipeline is designed to work in any domain in which repeating patterns are present. Nevertheless, if we were to use the pipeline only in one visual domain (e.g. woven fabrics), it should be worth exploring the possibility of fine-tuning the neural networks in the pipeline so that the filters learned by them are more closely related to that domain. This could be beneficial for both the repeating pattern size extraction and for the optimal tile synthesis. Finally, the network that was used to find the repeating pattern is a baseline AlexNet, which is not a state-of-the-art model in image classification problems. We chose this network due to its simplicity and widespread availability of pre-trained versions on-line. All the same, there are CNN models with less parameters and a similar level of accuracy [46] or more complex but with a richer set of filters, like the VGG-19 model we used for the perceptual loss calculation. Therefore, it should be possible to improve this algorithm by updating those steps with more recent deep learning models.

Acknowledgements. We thank the reviewers for their insightful comments and suggestions. Elena Garces was partially supported by a Juan de la Cierva - Formacion Fellowship from the Spanish Ministry of Science and Technology.

References

[1] Moritz, J, James, S, Haines, TS, Ritschel, T, Weyrich, T. Texture stationarization: Turning photos into tileable textures. In: Computer Graphics Forum (Proc. Eurographics); vol. 36. Wiley Online Library; 2017, p. 177–188.

[2] Liu, Y, Lin, WC, Hays, J. Near-regular texture analysis and manipulation. In: ACM Transactions on Graphics (Proc. SIGGRAPH); vol. 23. ACM; 2004, p. 368–376.

[3] Upchurch, P, Gardner, J, Pleiss, G, Pless, R, Snavely, N, Bala, K, et al. Deep feature interpolation for image content changes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017, p. 7064–7073.

[4] Lettry, L, Perdoch, M, Vanhoey, K, Van Gool, L. Repeated pattern detection using cnn activations. In: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV). IEEE; 2017, p. 47–55.

[5] Efros, AA, Freeman, WT. Image quilting for texture synthesis and transfer. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques. ACM; 2001, p. 341–346.

[6] Kaspar, A, Neubert, B, Lischinski, D, Pauly, M, Kopf, J. Self tuning texture optimization. In: Computer Graphics Forum; vol. 34. Wiley Online Library; 2015, p. 349–359.

[7] Gatys, L, Ecker, AS, Bethge, M. Texture synthesis using convolutional neural networks. In: Advances in neural information processing systems; vol. abs/1505.07376. 2015, p. 262–270.

[8] Sendik, O, Cohen-Or, D. Deep correlations for texture synthesis. In: ACM Transactions on Graphics; vol. 36. ACM; 2017, p. 161.

[9] Zhou, Y, Zhu, Z, Bai, X, Lischinski, D, Cohen-Or, D, Huang, H. Non-stationary texture synthesis by adversarial expansion. 2018,.

[10] Liu, Y, Collins, RT, Tsin, Y. A computational model for periodic pattern perception based on frieze and wallpaper groups. In: IEEE transactions on pattern analysis and machine intelligence; vol. 26. IEEE; 2004, p. 354–371.

[11] Zhao, P, Quan, L. Translation symmetry detection in a fronto-parallel view. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. IEEE; 2011, p. 1009–1016.

[12] Liu, J, Liu, Y. Grasp recurring patterns from a single view. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2013, p. 2003–2010.

[13] Pritts, J, Chum, O, Matas, J. Detection, rectification and segmentation of coplanar repeated patterns. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014, p. 2973–2980.

[14] Torii, A, Sivic, J, Pajdla, T, Okutomi, M. Visual place recognition with repetitive structures. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2013, p. 883–890.

[15] Lindeberg, T. Scale invariant feature transform 2012;7(5).

[16] Szeliski, R, Uyttendaele, M, Steedly, D. Fast poisson blending using multi-splines. In: 2011 IEEE International Conference on Computational Photography (ICCP). IEEE; 2011, p. 1–8.

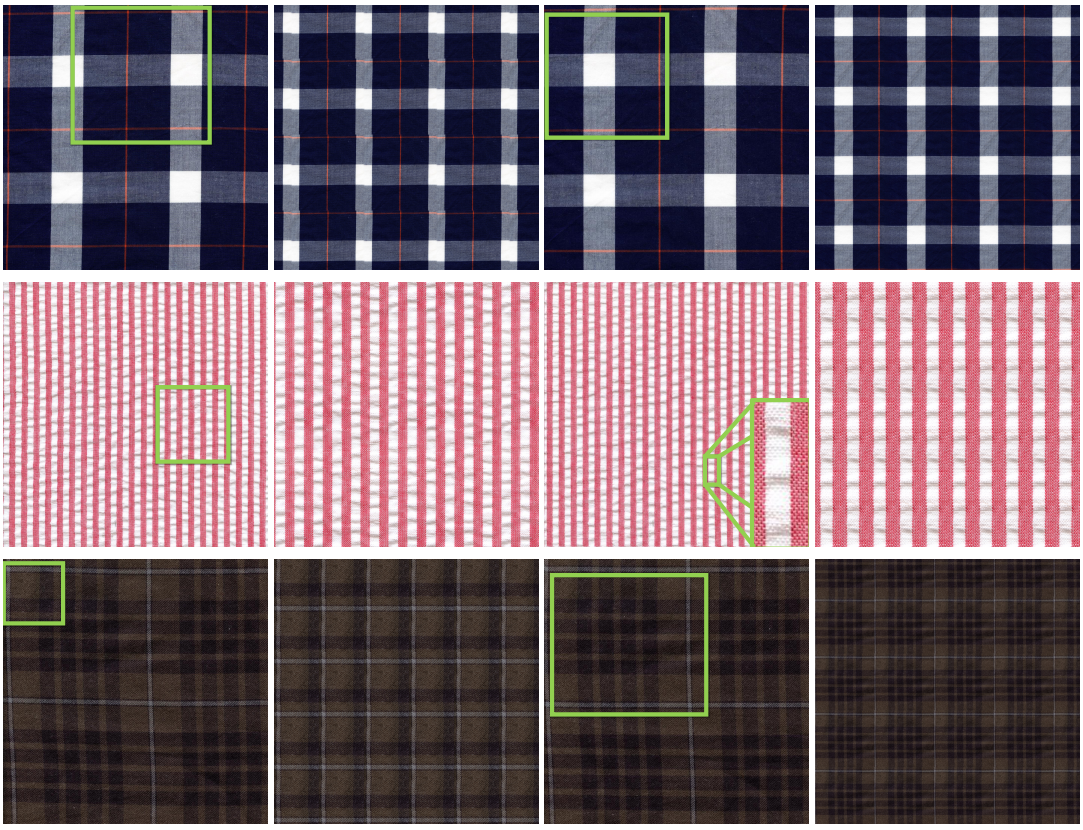

Fig. 7. Some results obtained by our method (ID images top-bottom #3, #4, #5). First and third column: The original input image and its regularized corresponding image, with the minimal tile segmented using a green box. Second and fourth column: The result of tiling the optimal tiles by taking as input the original and the regularized images, respectively. We can see that the result after regularization is more accurate and captures the optimal tile better than without it.

[17] Levin, A, Zomet, A, Peleg, S, Weiss, Y. Seamless image stitching in the gradient domain. In: European Conference on Computer Vision. Springer; 2004, p. 377–389.

[18] Adel, E, Elmogy, M, Elbakry, H. Image stitching system based on orb feature-based technique and compensation blending. In: International Journal of Advanced Computer Science and Applications; vol. 6. 2015,.

[19] Kim, VG, Lipman, Y, Funkhouser, TA. Symmetry-guided texture synthesis and manipulation. In: ACM Transactions on Graphics; vol. 31. 2012, p. 22–1.

[20] Weibel, T, Daul, C, Wolf, D, Rösch, R. Contrast-enhancing seam detection and blending using graph cuts. In: Proceedings of the 21st International Conference on Pattern Recognition. IEEE; 2012, p. 2732–2735.

[21] Darabi, S, Shechtman, E, Barnes, C, Goldman, DB, Sen, P. Image melding: Combining inconsistent images using patch-based synthesis. In: ACM Transactions on Graphics; vol. 31. 2012, p. 82–1.

[22] Dekel, T, Michaeli, T, Irani, M, Freeman, WT. Revealing and modifying non-local variations in a single image. In: ACM Transactions on Graphics; vol. 34. ACM; 2015, p. 227.

[23] Jafari-Khouzani, K, Soltanian-Zadeh, H. Radon transform orientation estimation for rotation invariant texture analysis. In: IEEE transactions on pattern analysis and machine intelligence; vol. 27. IEEE; 2005, p. 1004–1008.

[24] Zheng, D, Han, Y, Hu, JL. A new method for classification of woven structure for yarn-dyed fabric. In: Textile Research Journal; vol. 84. SAGE Publications Sage UK: London, England; 2014, p. 78–95.

[25] Guarnera, GC, Hall, P, Chesnais, A, Glencross, M. Woven Fabric Model Creation from a Single Image. In: ACM Transactions on Graphics; vol. 36. ACM; 2017, p. 1–13.

[26] Barnes, C, Shechtman, E, Goldman, DB, Finkelstein, A. The generalized patchmatch correspondence algorithm. In: European Conference on Computer Vision. Springer; 2010, p. 29–43.

[27] He, K, Zhang, X, Ren, S, Sun, J. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016, p. 770–778.

[28] Redmon, J, Divvala, S, Girshick, R, Farhadi, A. You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016, p. 779–788.

[29] Gatys, LA, Ecker, AS, Bethge, M. Image style transfer using convolutional neural networks. In: The IEEE Conference on Computer Vision and Pattern Recognition. 2016,.

[30] Mahendran, A, Vedaldi, A. Visualizing deep convolutional neural networks using natural pre-images. In: International Journal of Computer Vision; vol. 120. Springer; 2016, p. 233–255.

[31] Deng, G, Cahill, L. An adaptive gaussian filter for noise reduction and edge detection. In: 1993 IEEE Conference Record Nuclear Science Symposium and Medical Imaging Conference. IEEE; 1993, p. 1615–1619.

[32] Kwatra, V, Essa, I, Bobick, A, Kwatra, N. Texture optimization for example-based synthesis. In: ACM Transactions on Graphics; vol. 24. ACM; 2005, p. 795–802.

[33] Briechle, K, Hanebeck, UD. Template matching using fast normalized cross correlation. In: Optical Pattern Recognition XII; vol. 4387. International Society for Optics and Photonics; 2001, p. 95–103.

[34] Pérez, P, Gangnet, M, Blake, A. Poisson image editing. In: ACM Transactions on graphics); vol. 22. ACM; 2003, p. 313–318.

[35] Zhang, R, Isola, P, Efros, AA, Shechtman, E, Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018, p. 586–595.

[36] Sanakoyeu, A, Kotovenko, D, Lang, S, Ommer, B. A style-aware content loss for real-time hd style transfer. In: Proceedings of the European Conference on Computer Vision (ECCV). 2018, p. 698–714.

[37] Liu, Y, Zhang, YM, Zhang, XY, Liu, CL. Adaptive spatial pooling for image classification. In: Pattern Recognition; vol. 55. Elsevier; 2016, p. 58–67.

[38] Johnson, J, Alahi, A, Fei-Fei, L. Perceptual losses for real-time style

Fig. 8. Extracted optimal tiles in architectural elements and mosaics, and the result of tiling them. The images in these examples have not been regularized with NLV.

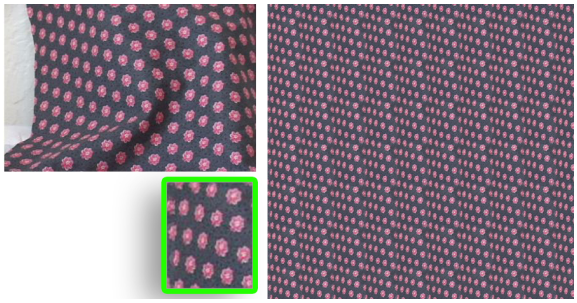

Fig. 9. Input image distorted geometrically. Top-left: input image. Bottom-left: extracted optimized tile. Right: Resulting tiled image.

transfer and super-resolution. In: European Conference on Computer Vision. 2016,.

[39] Krizhevsky, A, Sutskever, I, Hinton, GE. Imagenet classification with deep convolutional neural networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1. NIPS'12; USA: Curran Associates Inc.; 2012, p. 1097–1105.

[40] Deng, J, Dong, W, Socher, R, Li, LJ, Kai Li, , Li Fei-Fei, . ImageNet: A large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition. IEEE; 2009, p. 248–255.

[41] Paszke, A, Gross, S, Chintala, S, Chanan, G, Yang, E, DeVito, Z, et al. Automatic differentiation in pytorch. 2017,.

[42] Canziani, A, Paszke, A, Culurciello, E. An analysis of deep neural network models for practical applications. 2016,.

[43] Tomasi, C, Manduchi, R. Bilateral filtering for gray and color images. In: Iccv; vol. 98. 1998, p. 2.

[44] Müller, P, Zeng, G, Wonka, P, Van Gool, L. Image-based procedural modeling of facades. In: ACM Transactions on Graphics; vol. 26. New York, NY, USA: ACM; 2007,.

[45] Xiao, J, Fang, T, Zhao, P, Lhuillier, M, Quan, L. Image-based street-side city modeling. In: ACM Transactions on Graphics; vol. 28. New York, NY, USA: ACM; 2009, p. 114:1–114:12.

[46] Iandola, FN, Moskewicz, MW, Ashraf, K, Han, S, Dally, WJ, Keutzer, K. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. vol. abs/1602.07360. 2016,.

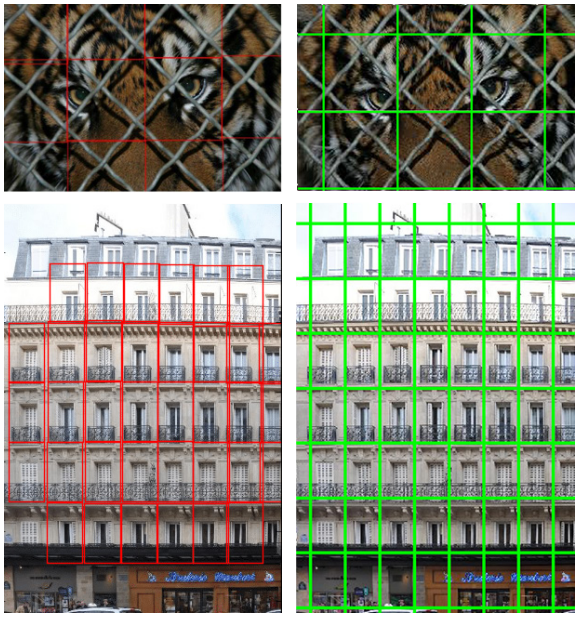

Fig. 10. Lattice extraction and comparison with Lettry's [4] work. Left: Theirs, as extracted from their paper. Right: Our results on the same images, using our simplification of their algorithm. It can be seen that both the tile size and the regions that were found are not significantly different, which shows that our modifications to their algorithm are not detrimental to its performance.

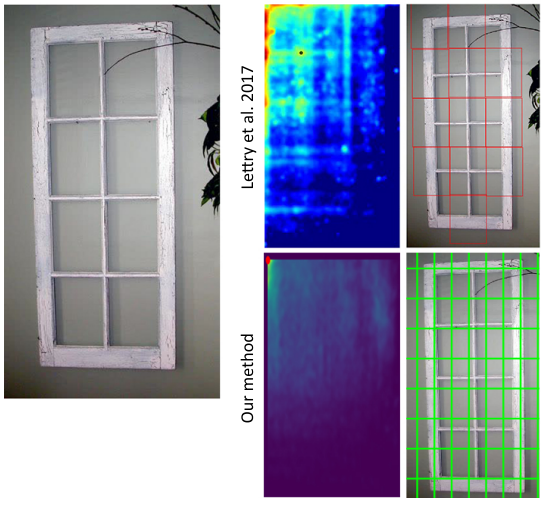

Fig. 11. Failure case for our method. Left: input images. Right-top: Lettry's voting space and lattice. Right-bottom: our voting space and lattice. Note that due to our simplification of the model or inappropriate parameters we do not handle textures with low frequency variability.