Noname manuscript No.
(will be inserted by the editor)

Augmenting Photographs with Textures Using the Laplacian Pyramid

Lars Doyle · David Mould

Received: date / Accepted: date

Abstract We introduce a method to stylize photographs with auxiliary textures, by means of the Laplacian pyramid. Laplacian pyramid coefficients from a synthetic texture are combined with the coefficients from the original image by means of a smooth maximum function. The final result is a stylized image which maintains the structural characteristics from the input, including edges, color, and existing texture, while enhancing the image with additional fine-scale details. Further, we extend patch-based texture synthesis to include a guidance channel so that texture structures are aligned with an orientation field, obtained through the image structure tensor.

Keywords Image Stylization · Laplacian Pyramid · Texture Synthesis

1 Introduction

Texture plays an important role in our appreciation and understanding of images. It can serve as a visual replacement for the tactile qualities that images lack, and prompt the photographed subject matter to appear more lively and interesting. Many photographers have realized this potential and have used textures to enhance their images. Often the intent is to transform an ordinary photograph into an artwork that contains characteristics of paintings, e.g., cracks, background materials, and brushstrokes. Other times, the intent is to create a "vintage" look, brought on by artificially weathering digital images. Figure 1 shows a photograph where an artist has used commercial image editing software to manually add texture.

Fig. 1 Manual image blending effects. Bill Showalter, *Barn at the end of the line*, 2017.

Alpha blending is the obvious way to combine a texture and an image. However, alpha blending has disadvantages: contrast is reduced, colors may be altered, and edges fade. Manual blending with per-pixel alpha is possible, but limited. In addition, it is often desirable for new textures to follow the orientation of the structures in the original image. For example, performing texture augmentation on a photo of a pet, an appealing option is to align the example texture with the fur orientation. Accomplishing this manually would be tedious and likely beyond the capabilities of many users.

Texture transfer is a related problem, but it completely replaces the original image with content from the texture. Conversely, in our approach, we want to preserve the edges and textures from the original image. In fact, the result of our method will exhibit the large-scale characteristics of the original, yet it will be

Lars Doyle
E-mail: larsdoyle@cmail.carleton.ca

David Mould
E-mail: mould@scs.carleton.ca

Fig. 2 Result from our texture augmentation method. Above: original; below: our result using $\alpha = 0.2$. Texture example is shown in the inset.

stylized with high-frequency details that are taken from an external texture source. These details could either be a simple background grain or any high-frequency texture that looks interesting when embedded within a photograph. Our method provides a novel tool that digital artists can use to create images that deliver a rich visual experience to the viewer. Figure 2 shows a result from our texture augmentation method. Notice that the outlines of the building remain crisp against the newly added texture, which is most visible in the low-frequency areas of the sky.

Contribution. We propose a novel artistic stylization method that augments an image with external textures. These textures are added with a method that preserves the edges, color, and textures from the original image. We have two major contributions:

1. We propose merging an image with a texture by mixing the coefficients of their Laplacian pyramid representations. The coefficient mixing uses the smooth maximum [11] function. Combining an image and a texture in this way retains the structural characteristics from the image while augmenting it with the fine-scale details of the texture.
2. We propose an irregular tiling based on SLIC superpixels [1] for patch-based texture synthesis. In the context of combining an image and a texture, the SLIC patches not only provide an irregular structure, making artifacts less noticeable, but also align with image edges, further concealing defects in the synthesized texture when it is integrated with the image.

2 Related Work

Image pyramids. Image pyramids are useful for manipulating and analyzing images at multiple scales. Broadly speaking, image pyramids are created by smoothing and downsampling the input image, creating a version half the size in both the horizontal and vertical dimensions, and then repeating until some stopping condition is reached – for example, the image is reduced to a single pixel in one of its dimensions. Gaussian smoothing yields the common Gaussian pyramid. Burt and Adelson introduced the Laplacian pyramid [8] whose levels consist of the difference images between successive levels in the Gaussian pyramid. Its coefficients represent the details and edges at different spatial scales of the input image. A variant of the image pyramid omits downsampling, yielding filtered images the same size as the original; in this case we refer to the resulting decomposition as an image stack.

While the original purpose of the Laplacian pyramid was image compression, it has been repurposed to include multi-resolution image blending [9], detail/tone manipulation [31], and style transfer [4,36]. For example, Shih et al. [36] rescale the individual coefficients of a Laplacian stack to transfer the characteristics of high-quality portrait photographs to casual snapshots. Other authors stylize images using an image decomposition based on the Bilateral filter [38]. This variant enables detail and texture coefficients to be manipulated independently from the edge coefficients, which are maintained in the higher levels of the stack. Bae et al. [5] base a style transfer method on a two-scale Bilateral stack. Similarly, Fattal et al. [18] propose detail enhancement using a multi-scale Bilateral stack. Their method combines the coefficients between multi-light image collections to obtain an enhanced composite image.

These methods that we mention here all seek a photorealistic look, either through detail and contrast enhancement or manipulating muti-scale features to match a target image. We extend this body of research to include non-photorealistic image stylization by using the Laplacian pyramid to add details that wouldn't necessarily exist in natural images.

Example-based texture synthesis. Our texture augmentation system requires, as input, a texture that has been synthesized to the dimensions of the input image. We employ non-parametric texture synthesis to this end. Non-parametric texture synthesis can be categorized into pixel-based [3,15,41], patch-based [14,29], or optimization-based [26,42] methods. The research in this field is extensive and we refer the reader to Wei et al.'s [40] comprehensive survey for detailed introductions to traditional techniques.

At the heart of all non-parametric methods is a nearest-neighbour search algorithm that selects the next pixel or patch in the synthesis process. Ashikhmin's coherence search [3] has been influential both in texture synthesis and in the *Patchmatch* [6] nearest-neighbour-field algorithm.

Texture transfer. Texture transfer uses a target image to guide texture synthesis. The output image is constructed from the texture exemplar but retains the large-scale configuration of the target. Efros and Freeman [14] augment their patch-based method to include luminance features and coerce a synthesized texture to appear like a target image. Related to our work, Lee et al. [28] synthesize oriented textures by adding gradient information to their neighbourhood similarity metric. Hertzmann et al. [22] frame texture transfer as a mapping between two sets of image pairs. An example pair represents a before-and-after transformation of an image. This transformation is learned and applied to an input image so that it appears to have undergone the same transformation. Okura et al. [30] employ a similar strategy to hallucinate scene changes in outdoor photography: observing that texture transfer can erroneously destroy image structure, they propose a method that shifts between colour transfer [37] and texture transfer to optimally represent the transformation. Fišer et al. [19] build upon optimization-based texture synthesis [26,42] to transfer texture from an example image to video using a series of guidance channels.

Other researchers have estimated 3D depth and surface orientation to drastically change the appearance of object surfaces in images. Fang et al. [17] induce foreshortening effects by warping texture patches to follow the surface orientation of an object. Khan et al. [25] extend beyond texture transfer, in editing material appearance, by allowing specularity and transparency editing.

Recently, advances to style transfer have been made possible through the use of convolutional neural networks [20,24]. These methods depart from the typical low-level features that are used in texture transfer and learn high-level semantic features that can represent more complex transformations.

Our work employs texture synthesis to generate image content, but has different objectives from previous work in texture transfer. We do not intend to synthesize a new image, but rather to enhance an existing one using fine-scale features that are extracted from texture examples.

Image Enhancement. Our system augments images with high frequency details and, as such, also contains similarities with super-resolution [13,24,34] and other forms of image enhancement [23]. However, in contrast to these methods, we seek a stylized look, as opposed to photorealism, and our output image is the same resolution as the input. Recent image enhancement methods center around convolutional neural networks (CNN), such that a low-resolution or corrupted image is passed through a transformation network to produce a high-resolution or high-quality output. Dong et al. [13] train a CNN using a per-pixel loss function to perform super-resolution. Other researchers have turned to perceptual loss functions that, while achieving lower PSNR scores than per-pixel loss, relate better to human perception. Johnson et al. [24] use feature loss extracted from the activation layers of a pre-trained classification network. Sajjadi et al. [34] add texture loss, computed through the Gram matrix [20] of a classification network, combined with adversarial training [21]. Ignatov et al. [23] use similar techniques – while training on mobile-DSLR photograph pairs – to transform low quality inputs into higher-quality outputs.

3 Enhancing images with texture exemplars

Figure 3 illustrates the pipeline of our system. The pipeline contains two phases: a texture synthesis phase, where we generate texture content, and a texture augmentation phase, where fine-scale features of this texture are merged with the input image. The merging is done by computing the Laplacian pyramid of both the image and the texture, and using the *smooth maximum* [11] function to combine individual Laplacian coefficients; we reconstruct the final image from the resulting combined pyramid. Details are given in the following subsections.

3.1 Texture synthesis

Our texture augmentation system requires two inputs: an input image and a texture image. In this section we describe our method for synthesizing the texture

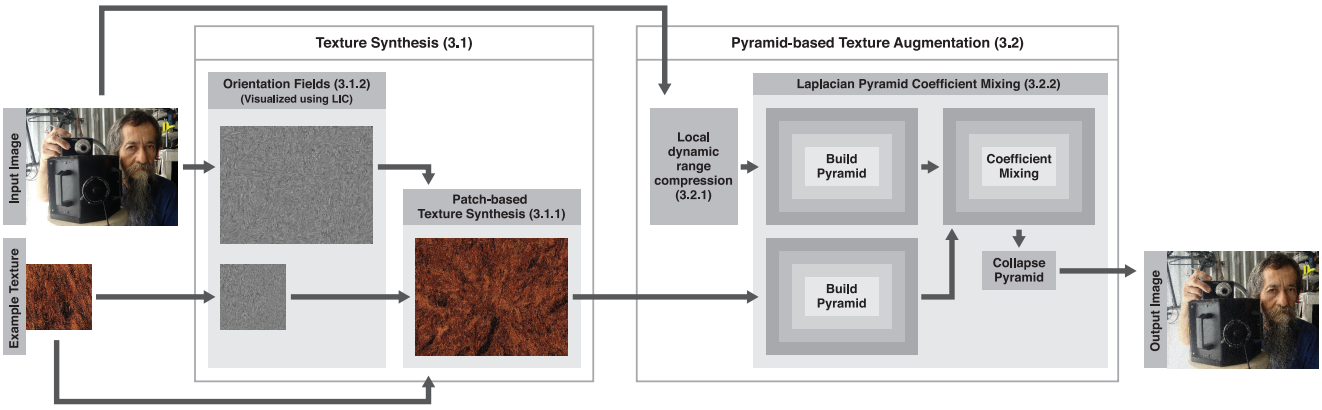

Fig. 3 The pipeline of our texture augmentation system.

image, a patch-based system in the tradition of Image Quilting [14]. We first describe our basic method (Section 3.1.1) and then discuss extending it to include a guidance channel (Section 3.1.2). The guide image will control the orientation of texture structures in the output.

However, our pyramid-based texture augmentation method (Section 3.2) does not depend on any particular texture synthesis method. We can use any synthesis method [19,43] where local feature orientation can be controlled, or even just resort to static textures.

3.1.1 Patch-based texture synthesis

The original Image Quilting algorithm places texture patches on a regular grid. Unfortunately, the repeating grid pattern can make any imperfections in the synthesized texture easily noticeable. Using irregular texture patches [27] is one strategy to hide these imperfections. By avoiding predictable patch placements, poorly matched seams can go unnoticed for many near-stochastic textures. We divide the output image plane into irregular patches by applying *Simple Linear Iterative Clustering* (SLIC) [1] to the input image. Besides providing an irregular tiling, SLIC also aligns patch boundaries with image edges; this alignment sometimes conceals poorly matched seams in the synthesized texture, since they will be hidden by the image edges in the final integrated image.

We make one addition to the standard SLIC algorithm. Since super-pixels in flat featureless image regions tend to have regular shapes, we augment SLIC's distance metric to include an additional term that uses color information from an outside guidance image. The distance metric then becomes:

$$D = \lambda d_{guide} + (1 - \lambda) d_{lab} + \frac{m}{S} d_{xy}, \quad (1)$$

where d_{guide} is the L_2 norm between pixels and their cluster centres in the guide. The parameter λ controls the influence of the guide image. Typically, we use $\lambda = 0.2$, a value small enough that its effect is only felt when d_{lab} is low. Figure 4 shows the result of our super-pixel segmentation with and without the use of a guide image; here, the guide image is a sample of Perlin noise.

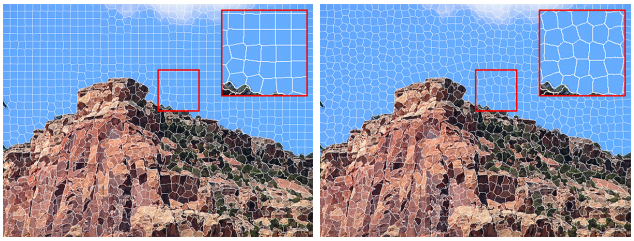

Fig. 4 SLIC patches. Left: original distance metric; right: our distance metric.

Our patch search method is inspired both by Ashikhmin's pixel-based coherence search [3] and by the random search method used in PatchMatch [6]. Given a patch that we are about to synthesize, we consider all adjacent patches in the previously determined texture. Extending these patches into the current location gives us an initial set of candidates which are then evaluated with the *sum-of-squared-differences* on the patch overlap. Choosing the best match from the coherence search, we then proceed to the random sampling method from PatchMatch attempting to improve upon this initial selection.

The final step is to find an optimal seam through the region where adjacent patches overlap. We define $error(u)$ as the squared difference between the previously determined texture and the current texture patch at location u. A min-cut is then obtained using Dijkstra's algorithm on the pixel graph using $max(error(u), error(v))$ as the edge weights between pixels u and v.

3.1.2 Oriented texture synthesis

We intend to synthesize textures so that local structures are oriented similarly to corresponding locations in the input image. Local image orientation can be described through the image structure tensor [7]. Its eigendecomposition gives eigenvalues $\lambda_1 \gg \lambda_2$ and their corresponding eigenvectors $\mathbf{e}_1$ and $\mathbf{e}_2$. The eigenvector $\mathbf{e}_1$ points in the direction of the image gradient. However, the structure tensor field can be noisy and will benefit from smoothing. Criminisi et al. [12] present a method for edge-aware color smoothing in images using a geodesic-distance transform. We would like our orientation fields to possess edge-aware smoothness: to wit, orientation should change gradually within similar regions, but can change drastically across edges. Criminisi et al.'s method can naturally be adapted to our purposes, averaging tensors rather than colors. This smoothing process also allows tensors with arbitrary orientation to receive influence from nearby structures.

In computing the geodesic-distance transform we define the distance between any two tensors as:

$$D = w_{xy}d_{xy} + w_{lab}d_{lab} + w_{or}d_{or}, \tag{2}$$

where d_{xy} and d_{lab} are the spatial and color-space L_2 norm and d_{or} is tensor orientation distance. Each component is weighted as w_{xy}, w_{lab}, and w_{or}, respectively. Following Akl et al. [2] we compute d_{or}, between tensors M_1 and M_2 as:

$$d_{or}(M_1, M_2) = |sin(\theta_1 - \theta_2)| \times min(C_1, C_2), \tag{3}$$

using $\theta = tan^{-1}(\mathbf{e}_{1y}/\mathbf{e}_{1x})$ to determine tensor orientation. The orientation coherence is computed as:

$$C = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2 + K}, \tag{4}$$

where K is a constant that both prevents division by zero and lowers the coherence measure of weak structures. We set K to 0.0015 in our examples. Distance terms d_{lab} and d_{lab} are normalized by dividing them by the 95^{th} percentile of observed distance values in the appropriate domain. Finally, in all of our results we set the weight terms w_{xy}, w_{lab}, and w_{or} to 1, 3, and 5, respectively.

We produce an orientation guide for both the input image and the texture example. At each pixel location we extract a vector indicating the local image orientation, using the eigendecomposition of the smoothed structure tensor. We rotate texture patches so as to align the orientations of the auxiliary texture and the input image. A simple update incorporates orientation into the method presented in Section 3.1.1. We now rotate each texture patch so that orientations are aligned between the patch centre and a corresponding position in the underlying image. This rotation is done before calculating the overlap error on a given candidate. A sample result from this process is shown in Figure 5.

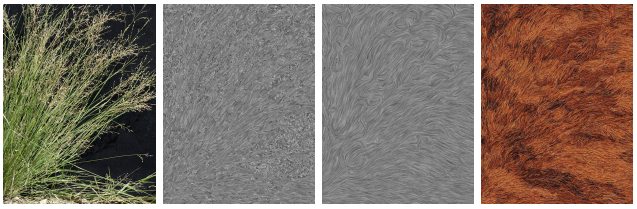

Fig. 5 Oriented texture synthesis. Left to right: input image, original structure tensor, smooth structure tensor, synthesized texture. Orientation fields are visualized using LIC [10].

3.2 Pyramid-based texture augmentation

Having now synthesized a texture, we combine it with the input image using the Laplacian pyramid. We give details on our Laplacian pyramid calculations in Section 3.2.2. However, because the resulting pyramid may produce out-of-scale intensity values, we first compress extreme luminance values in a pre-processing step in order to minimize saturation in highlight and shadow regions; this is described first in Section 3.2.1.

3.2.1 Local dynamic range compression

By altering the Laplacian coefficients in the image pyramid, pixel values may extend beyond the legal $[0, 1]$ intensity value interval. A simple strategy to fix this problem is to clamp invalid pixel values. However, this will only allow us to represent part of the new texture since many coefficients will be thrown away. Rescaling the original image is not an appealing option either since that will produce a flat, lower-contrast result. We intend to maintain the original dynamic range of the input image as much as possible. We use the method of Pérez et al. [32] for local dynamic range compression in order to selectively rescale highlight and shadow regions. By limiting the dynamic range compression method to selected regions, we preserve most of the image while creating more space in the high and low end of the intensity range. This process involves two steps: (1) We present our automatic method for selecting image regions to be compressed. (2) We apply Pérez et al.'s local dynamic range compression method to each selected region.

Step 1. Region Selection: The following presentation treats only the high end of the intensity range, but the low end of the intensity range is symmetric.

We use hysteresis thresholding to select bright image regions for dynamic range compression. Given a user-defined pair of threshold values, t_{high} and t_{low}, we select contiguous regions in the image where image intensity values exceed t_{low} while also containing pixels which exceed t_{high}. The intent of using two threshold values is twofold: the upper threshold marks the true highlights that we want to compress, while the lower threshold marks a wider region around the selected highlight within which we interpolate the image intensity. In our examples we set t_{high} to 0.96 and t_{low} to 0.82 for bright regions; in dark regions, $t_{low} = 0.04$ and $t_{high} = 0.18$.

One might alternately imagine choosing the larger region spatially, using a mechanism such as morphological dilation. However, where bright image regions exist adjacent to strong step edges, we would like to prevent the larger region from crossing these boundaries. Our hysteresis thresholding method accomplishes this and avoids destructively editing the profiles of important edges through dynamic range compression.

Working in the luminosity channel of CIELAB color space, we first threshold the input image at t_{low}. Next, we obtain the connected components of the resulting binary image. These regions will be candidates for dynamic range compression provided that their pixel count exceeds a user-specified amount ϵ; we used $\epsilon = 20$ for 1.5 megapixel images in all examples. Next, we discard any candidate regions that lack true highlights, i.e., those that do not contain pixels with intensity above t_{high}. Figure 6 provides a visualization of this selection process in the top row. Pixels marked at the lower threshold are shown in yellow and pixels marked at the upper threshold are shown in red. In the bottom row of this figure we show a texture augmented image with (left) and without (right) local dynamic range compression. Notice how the input texture is better represented against bright background in the right image.

Step 2. Dynamic Range Compression: We apply dynamic range compression to the selected regions. Following Pérez et al., we use the following remapping function to modify the original image gradient and guide Poisson blending:

$$G' = sign(G) \times \alpha^{\beta} \times |G|^{1-\beta}, \tag{5}$$

where G is the gradient vector field of the input image. We set α to 0.2 times the average gradient magnitude within the selected region, and β to 0.2, following Pérez et al.

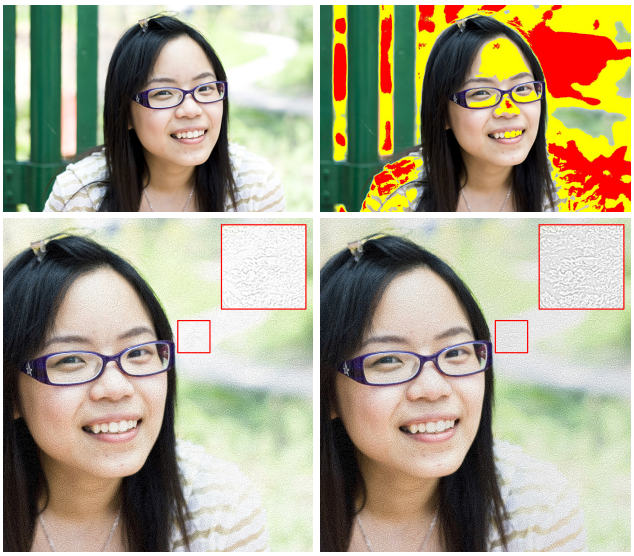

Fig. 6 Region selection is illustrated in the top row. Top left: original image; top right: upper threshold shown in red, lower threshold shown in yellow. The yellow regions indicate the final selections provided that they enclose red pixels. The effect of compressing extreme values is shown in the bottom row. Bottom left: without compression; bottom right: extremes compressed. Both results set $\alpha = 0.1$.

3.2.2 Laplacian pyramid coefficient mixing

Given two Laplacian pyramids L^I and L^T, derived from an input and texture image, we compute a result pyramid L^R that combines coefficients from both inputs. A simple strategy is to compare the absolute value between coefficients from both inputs and choose the maximum while retaining its sign. This is similar to the strategy used by Pérez et al. [32] where they mix image gradients to blend images. However, this "choose the maximum" rule can cause image edges to become hidden amongst newly added textures. Instead, we suggest using the *smooth maximum* [11]. The smooth maximum of input values u and v is computed as follows:

$$SM(u, v, k) = ln\big(exp(ku) + exp(kv) - 1\big)\frac{1}{k}, \tag{6}$$

where k is a parameter controlling the degree of smoothness in the maximum function. For large values of k, Equation 6 degrades to an ordinary maximum function. Smaller values of k, in contrast, yield an output value above the larger of u and v. However, this amplification of the output value is not distributed equally. Instead, where u and v are dissimilar, their smooth maximum will approach the maximum. Alternately, where u and v are equal the output will receive the largest amplification. Hence, we can intuitively visualize a maximum function that rounds off the corner where u and v approach the same value. Note that we also subtract a one from the sum of the exponentiated inputs. This adjust-

ment ensures that when both Laplacian coefficients are zero then the output will also be zero.

Using Equation 6, we compute each output coefficient $L_l^R(x,y)$ at level l as follows:

$$L_l^R(x,y) = \Psi \times SM\big(|L_l^I(x,y)|, w_l|L_l^T(x,y)|, k\big), \qquad (7)$$

where the variable Ψ can be 1 or -1 and refers to the sign of the coefficient with the larger absolute value. We chose a value of k designed to amplify the output in regions where texture and input images have similar Laplacian coefficient magnitudes, i.e., where auxiliary texture and image details align. A slight contrast enhancement in these regions allow the original image structure to remain salient against a baseline with increased texture activity. Experimentally, we set k to 25 assuming pixel values in the interval $[0,1]$. In Figure 7 we can see that this parameter setting, observed in the second image from the right, gives a slight boost to the Laplacian coefficients; effectively increasing contrast.

Fig. 7 Varying the parameter k in Equation 7. From left to right: original image, $k = 500$, $k = 25$, and $k = 1$. All results set $\alpha = 0.15$.

In addition, rather than letting the example texture completely dictate the look of the resulting texturized image, we control the texture's influence using parameter w_l:

$$w_l = \frac{\alpha}{\phi_l \times 2^l}, \qquad (8)$$

where α is a user-defined term that controls the intensity of the texture coefficients and ϕ_l is a normalization term that adjusts for the contrast of the texture exemplar. To control for outliers, we set ϕ_l to the 95^{th} percentile of the Laplacian coefficients observed at level l in the texture image. Further, since we intend texture to be incorporated mostly into the finer-scale pyramid levels, we scale down the texture coefficients by 2^l: thus, for any texture, only the fine-scale features are retained.

The parameter α in Equation 8 provides a predictable means to control the intensity of the added textures. Figure 8 illustrates different texturizing effects at $\alpha = 0.04$, 0.1, and 0.2. The lowest setting introduces texture so subtly that it is barely perceptible. At the highest setting, the added texture is emerging against the high contrast image details at the front of the car. This has probably pushed it too far but the preferred outcome is, of course, up to the user.

Fig. 8 Varying the texture intensity parameter α (Equation 8). Top left: input image; top right $\alpha = 0.04$; bottom left: $\alpha = 0.1$; bottom right: $\alpha = 0.2$.

3.3 Masking effects

In many cases, the results can be more interesting if we allow the user to control the texture augmentation. A typical use case involves a digital artist selecting targeted regions, then adding texture independently to different image locations: for example, one texture could be applied to the foreground while another is applied to the background. We use the GrabCut [33] semi-automated segmentation method to define a collection of masked regions on our input images. Manually segmenting an image takes roughly 30–60 seconds.

Figure 9 shows an example where we add texture to targeted regions of an image. Adding textures to part of an image can be used as a general technique to emphasize a particular subject. The image of the parrot contains two added textures, one in the foreground and one in the background. Using two textures in the manner often helps to differentiate objects. We can also vary the texture intensity for each region independently. The background texture is applied more subtly to the forested area while the foreground texture is more obvious on the parrot.

Another use of the structure tensor that was introduced in Section 3.1.2 is to use it to build coherence maps. Orientation coherence (Equation 4) is measured by the relative strength of the tensor's dominant eigenvalue. We smooth the output of Equation 4 with a cross-bilateral filter, guided by the original input image. This process provides us a good predictor for how well directional texture synthesis will represent the underlying image content and we can use this information to blend isotropic and anisotropic textures into image regions where coherence is low or high, respectively. Isotropic textures are synthesized without rotating patches and anisotropic textures are synthesized using patch rota-

Fig. 9 Result with a user defined mask. Above: input image (texture and mask shown in insets); below: result using $\alpha = 0.1$ in the background and $\alpha = 0.2$ on the parrot.

tions. The coherence map contains entries in the interval $m = [0, 1]$ which are used to obtain a blended texture Laplacian pyramid where individual coefficients are calculated as:

$$L_l^T(x,y) = m(x,y)L_l^{T1}(x,y) + (1 - m(x,y))L_l^{T2}(x,y), \quad (9)$$

where L_l^{T1} and L_l^{T2} are Laplacian pyramids for the anisotropic and isotropic texture images, respectively. This blended Laplacian pyramid will be used as the input for coefficient mixing in Equation 7.

Figure 10 shows the result of blending two tree-bark textures – showing isotropic and anisotropic qualities – in the top and bottom texture samples, respectively. In the flat regions of the skin, an isotropic bark texture dominates. Towards the jawline, the texture smoothly transitions to an anisotropic texture following the contours of the face. Texture in the jewellery is also mostly taken from the anisotropic source where it reinforces both linear and circular structures.

Fig. 10 Results using orientation coherence for a mask. Textures and masks are shown at top left. The isotropic texture (top example) uses $\alpha = 0.1$ and the anisotropic texture (bottom example) uses $\alpha = 0.2$.

4 Results and Discussion

By incorporating high-frequency stochastic textures into an image, we can increase the visual richness of otherwise flat image regions. The appearance of film grain in classic photography once provided similar characteristics. Other, highly stylized effects are created by coercing textures to follow an orientation field. For example, painterly effects can be produced, where textures follow edge and texture orientations.

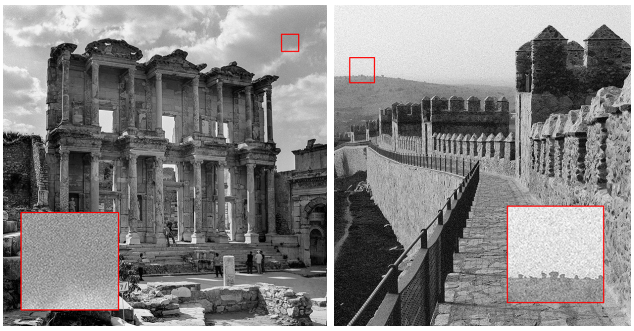

Fig. 11 Adding film grain to an image. Left: b&w film photograph; right: blending a scanned 35mm negative with a digital image using our method using $\alpha = 0.15$.

We present some results from our method in Figures 11 and 12. These images use a single texture, applied uniformly across the image plane. Figure 11 adds film grain to a greyscale digital image while Fig-

ure 12 incorporates textures with unambiguous orientation. On casual inspection, the augmented images do not appear drastically different from the input images. This is due to three reasons. First, our method changes Laplacian coefficients only in the luminosity channel. The color palette of the input image is thus retained. Second, our method maintains the dominant edges from the input. Finally, because our method reduces the effect of the texture coefficients at higher pyramid levels, the method maintains the large-scale tonal variation of the input.

All of the above characteristics are deliberate. Texture should be added where it is not already present; where there is significant texture or edges in the input image, the effect should be minimal. In this sense, our texture augmentation method is purely additive: textures can be added to an image but not removed. In flat or lightly textured regions, the smooth maximum function from Equation 6 favours Laplacian coefficients from the example texture. In highly textured regions, or at strong edges, the Laplacian coefficients from the input image are favoured. Observe the man's face in Figure 12. The synthesized texture follows the contours in his face and is seen clearly. Yet, prominent features, such as the eyes and facial creases, are still just as visible as they were in the original.

In the second image, we can clearly see the effect of the directional texture synthesis. The texture seems to flow around the berries and leaves, producing an illusion of movement. Many NPR painting styles have sought a similar goal in emulating highly visible and energetic brushwork, often mentioning Vincent van Gogh as inspiration [16,39].

4.1 Comparison with related work

Figure 13 shows a comparison between our method and image blending modes that are found in commercial software. In order to be consistent with our method, we constrain blending to the luminosity channel. The alpha-blended image appears flat, since intensity values are averaged between the two inputs. The multiply and overlay blends, as found in Adobe® Photoshop®, are often used for manual texture blending. As we can see, these modes also produce color distortions which may be unintended. In contrast, our method maintains color and edges. The type of texture shown here is difficult to synthesize, owing to its non-stationary features; images such as this are usually manually created. We used our Laplacian pyramid method to add the static texture to the photograph; the result is far superior to image blending, and could be used directly in some cases. Still, there is a visual separation between the photograph and the added texture, with extended texture features crossing over image edges and signaling the viewer that the texture is a different layer. Next consider the last result, combining the photograph with a texture synthesized from a small sample of the exemplar. The texture synthesis method is not able to reproduce the large-scale structures that help characterize this texture. However, the synthetic texture better integrates with the photograph: texture features no longer cross image edges, since they are oriented parallel with them. This results in a more unified appearance between the added texture and the original content.

Figure 14 gives a comparison between our texture augmentation method and Semmo et al.'s oil painting filter [35]. Although their main contribution is the image abstraction effect, we share their goal of adding texture to the surface of an image. Their textures are created by applying lighting to a heightfield derived from image orientation, thus producing a painterly effect. Our method uses example-based textures and is more versatile; also, our method more thoroughly integrates the texture into the input photograph. In our example at the bottom, small-scale details in the foreground are better preserved. While we apply textures to photographic images, a possible future direction could use our approach in conjunction with a more severe stylization of the input photograph.

The Textureshop system of Fang et al. [17] goes further than we do in attempting to realistically apply texture to 3D object surfaces. In our work, we focus on automatically aligning textures with feature orientations. This is only a minimal concern in Textureshop. They instead allow a user to manually indicate texture orientation. Even then, orientation is on a much larger scale than we have locally defined it. The bulk of their efforts are, instead, focused on distorting the example texture to indicate the correct perspective of the target object. In the example, shown at the top of Figure 15, it appears that the lion sculpture is composed of the texture substance. In our approach, which can be seen below, it appears that the texture has been etched into the surface of the stone.

4.2 Limitations

Extending our method to operate on three color channels would not be possible. The smooth maximum function operates independently at each image location, level by level, in the Laplacian pyramid. This process is effective for luminosity changes, allowing new texture coefficients to build upon the lower pyramid levels. However, by extending to all color channels the results

Fig. 12 Results with a single texture. Both examples set $\alpha = 0.1$.

Fig. 13 Comparison with image blending. Top: input image (left) and static texture (right); middle: image blending constrained to the luminosity channel, alpha-blend (left), Photoshop's overlay-blend (center) and multiply-blend (right); bottom: our result using static texture (left), our result using texture synthesis (right). Bottom examples set $\alpha = 0.2$.

would be unpredictable, both within a channel and especially across channels.

Our method is designed to transfer high-frequency details from a texture example into an input image. However, many textures are characterized by lower-frequency variation. This class of texture presents a

Fig. 14 Comparison with Semmo et al.'s oil painting filter [35]. Above: Semmo et al.; below: ours using $\alpha = 0.2$.

Fig. 15 Comparison with Fang et al.'s Textureshop [17] method. Above: Fang et al.; below: ours using $\alpha = 0.15$.

problem for our method. For example, picture a large-scale checkerboard texture; only the Laplacian coefficients representing region edges will be transferred to the input image. As such, our method is limited to the high-frequency textures that we have used to demonstrate our method in this paper.

5 Conclusion

We presented two distinct image processing techniques and combined them to stylize photographs by adding textures. The first technique uses the Laplacian pyramid to mix coefficients between an input image and a synthesized texture. Using our method, we can produce image stylization effects that augment images with new textures while preserving strong edges, color, and original textures. Additionally, we draw on previous work in patch-based texture synthesis and develop a method for synthesizing textures to follow an orientation field.

We use a smooth maximum function to introduce new texture coefficients into the Laplacian pyramid of an image. This function differs from the ordinary maximum in that it amplifies similar values. This amplification helps to preserve edges against a background of newly added texture. Our method builds upon the original large-scale features of the input image by replacing coefficients in the high-frequency bands of the Laplacian pyramid. Adding texture in this fashion enhances the illusion that added textures naturally belong to the original image. Conversely, previous methods have sought to replace image content with new textures entirely.

We used SLIC-based patches as atomic units of texture, benefitting the synthesis process in two ways. First, they place seam imperfections in unpredictable locations; the irregular placements are less apparent than is a regular pattern. Second, used for combining photographs and textures, the SLIC patches tend to align patch boundaries with image edges. Since Laplacian coefficients from the texture are less likely to be transferred to the output in these locations, artifacts at these seams are less visible in the result.

Future Work. While we have designed a method to transfer characteristics from example textures into an image, we see several avenues to extend our work. Drawing inspiration from Textureshop [17], we would like to enhance the realism of our added textures by considering 3D aspects. Using photometric methods from computer vision we can estimate the surface orientation of objects in the input image. We could use this information to adjust the magnitude and spatial frequency of texture features to create perspective effects. We also suggest that demands on the user could be minimized with adaptations to the GrabCut semi-automatic segmentation system. We believe that augmenting color information with texture descriptors could lead to more accurate results in our application and reduce the need for iterative user involvement. This would free the user for higher-level tasks such as experimenting with different textures that can be used to enhance digital artworks.

Acknowledgements We would like to thank the anonymous reviewers for many insightful comments. Thanks also to Eric Paquette and members of the Graphics, Imaging and Games Lab for productive comments and discussions. Funding for this work was provided by NSERC and by Carleton University.

We used many images from Flickr under a Creative Commons license. Thanks to the numerous photographers who provided material: Bill Showalter (barn), Jim Sorbie (cabin), Alana Sise (rocky hill), Sérgio Sakakibara (cameraman), Harry Rose (grass), delta ! (girl), bananaana04 (parrot), Maria Morri (earring), Donald Lammers (ruin), Tim Adams (walkway), Ryan Basilio (berries), Gábor Lengyel (portrait), Stephanie Kroos (lion), and Martin Pettitt (car).

References

1. Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Susstrunk, S.: SLIC superpixels compared to state-of-the-art superpixel methods. IEEE Transactions on Pattern Analysis and Machine Intelligence **34**(11), 2274–2282 (2012)
2. Akl, A., Yaacoub, C., Donias, M., Da Costa, J.P., Germain, C.: Texture synthesis using the structure tensor. IEEE Transactions on Image Processing pp. 4082–4095 (2015)
3. Ashikhmin, M.: Synthesizing natural textures. In: Proceedings of the 2001 Symposium on Interactive 3D Graphics, I3D '01, pp. 217–226. ACM, New York, NY, USA (2001)
4. Aubry, M., Paris, S., Hasinoff, S.W., Kautz, J., Durand, F.: Fast local laplacian filters: Theory and applications. ACM Trans. Graph. **33**(5), 167:1–167:14 (2014)
5. Bae, S., Paris, S., Durand, F.: Two-scale tone management for photographic look. ACM Trans. Graph. **25**(3), 637–645 (2006)
6. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.: Patchmatch: a randomized correspondence algorithm for structural image editing. ACM Trans. Graph. **28**(3), 24 (2009)
7. Brox, T., van den Boomgaard, R., Lauze, F., van de Weijer, J., Weickert, J., Mrázek, P., Kornprobst, P.: Adaptive structure tensors and their applications. In: J. Weickert, H. Hagen (eds.) Visualization and Processing of Tensor Fields. Mathematics and Visualization, chap. 2, pp. 17–47. Springer, Berlin, Heidelberg (2006)
8. Burt, P., Adelson, E.: The laplacian pyramid as a compact image code. IEEE Transactions on Communications **31**(4), 532–540 (1983)
9. Burt, P.J., Adelson, E.H.: A multiresolution spline with application to image mosaics. ACM Trans. Graph. **2**(4), 217–236 (1983)
10. Cabral, B., Leedom, L.C.: Imaging vector fields using line integral convolution. In: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '93, pp. 263–270. ACM, New York, NY, USA (1993)
11. Cook, J.: Basic properties of the soft maximum. UT MD Anderson Cancer Center Department of Biostatistics Working Paper Series. Working Paper 70 (2011). URL http://biostats.bepress.com/cgi/viewcontent.cgi?article=1073&context=mdandersonbiostat
12. Criminisi, A., Sharp, T., Rother, C., Pérez, P.: Geodesic image and video editing. ACM Trans. Graph. **29**(5), 134:1–134:15 (2010)
13. Dong, C., Loy, C.C., He, K., Tang, X.: Image super-resolution using deep convolutional networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **38**(2), 295–307 (2016)
14. Efros, A.A., Freeman, W.T.: Image quilting for texture synthesis and transfer. In: Proceedings of SIGGRAPH 98, SIGGRAPH '01, pp. 341–346. ACM, New York, NY, USA (2001)
15. Efros, A.A., Leung, T.K.: Texture synthesis by non-parametric sampling. In: Proceedings of the International Conference on Computer Vision, *ICCV '99*, vol. 2, pp. 1033–. IEEE Computer Society, Washington, DC, USA (1999)
16. Elad, M., Milanfar, P.: Style transfer via texture synthesis. IEEE Transactions on Image Processing **26**(5), 2338–2351 (2017). DOI 10.1109/TIP.2017.2678168
17. Fang, H., Hart, J.C.: Textureshop: texture synthesis as a photograph editing tool. ACM Trans. Graph. **23**(3), 354–359 (2004)
18. Fattal, R., Agrawala, M., Rusinkiewicz, S.: Multiscale shape and detail enhancement from multi-light image collections. ACM Trans. Graph. **26**(3) (2007)
19. Fišer, J., Jamriška, O., Simons, D., Shechtman, E., Lu, J., Asente, P., Lukáč, M., Sýkora, D.: Example-based synthesis of stylized facial animations. ACM Trans. Graph. **36**(4), 155:1–155:11 (2017)
20. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2414–2423 (2016)
21. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, K.Q. Weinberger (eds.) Advances in Neural Information Processing Systems 27, pp. 2672–2680. Curran Associates, Inc. (2014)
22. Hertzmann, A., Jacobs, C.E., Oliver, N., Curless, B., Salesin, D.H.: Image analogies. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '01, pp. 327–340. ACM, New York, NY, USA (2001)
23. Ignatov, A., Kobyshev, N., Timofte, R., Vanhoey, K., Van Gool, L.: Dslr-quality photos on mobile devices with deep convolutional networks. In: The IEEE International Conference on Computer Vision (ICCV) (2017)
24. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: B. Leibe, J. Matas, N. Sebe, M. Welling (eds.) Computer Vision – ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II, pp. 694–711. Springer International Publishing, Cham (2016)
25. Khan, E.A., Reinhard, E., Fleming, R.W., Bülthoff, H.H.: Image-based material editing. ACM Trans. Graph. **25**(3), 654–663 (2006)
26. Kwatra, V., Essa, I., Bobick, A., Kwatra, N.: Texture optimization for example-based synthesis. ACM Trans. Graph. **24**(3), 795–802 (2005)
27. Kwatra, V., Schödl, A., Essa, I., Turk, G., Bobick, A.: Graphcut textures: Image and video synthesis using graph cuts. ACM Trans. Graph. **22**(3), 277–286 (2003)
28. Lee, H., Seo, S., Yoon, K.: Extended papers from npar 2010: Directional texture transfer with edge enhancement. Comput. Graph. **35**(1), 81–91 (2011)
29. Liang, L., Liu, C., Xu, Y.Q., Guo, B., Shum, H.Y.: Real-time texture synthesis by patch-based sampling. ACM Trans. Graph. **20**(3), 127–150 (2001)
30. Okura, F., Vanhoey, K., Bousseau, A., Efros, A.A., Drettakis, G.: Unifying color and texture transfer for predictive appearance manipulation. Computer Graphics Forum **34**(4), 53–63 (2015)
31. Paris, S., Hasinoff, S.W., Kautz, J.: Local laplacian filters: Edge-aware image processing with a laplacian pyramid. ACM Trans. Graph. **30**(4), 68:1–68:12 (2011)
32. Pérez, P., Gangnet, M., Blake, A.: Poisson image editing. In: ACM Trans. Graph., vol. 22, pp. 313–318. ACM (2003)
33. Rother, C., Kolmogorov, V., Blake, A.: Grabcut: Interactive foreground extraction using iterated graph cuts. ACM Trans. Graph. **23**(3), 309–314 (2004)
34. Sajjadi, M.S., Schölkopf, B., Hirsch, M.: Enhancenet: Single image super-resolution through automated texture synthesis. arXiv preprint arXiv:1612.07919 (2016)

35. Semmo, A., Limberger, D., Kyprianidis, J.E., Döllner, J.: Image stylization by oil paint filtering using color palettes. In: Proceedings of the Workshop on Computational Aesthetics, CAE '15, pp. 149–158. Eurographics Association, Goslar Germany, Germany (2015)
36. Shih, Y., Paris, S., Barnes, C., Freeman, W.T., Durand, F.: Style transfer for headshot portraits. ACM Trans. Graph. **33**(4), 148:1–148:14 (2014)
37. Shih, Y., Paris, S., Durand, F., Freeman, W.T.: Data-driven hallucination of different times of day from a single outdoor photo. ACM Trans. Graph. **32**(6), 200:1–200:11 (2013)
38. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: Computer Vision, pp. 839–846. IEEE (1998)
39. Wang, B., Wang, W., Yang, H., Sun, J.: Efficient example-based painting and synthesis of 2d directional texture. IEEE Transactions on Visualization and Computer Graphics **10**(3), 266–277 (2004)
40. Wei, L.Y., Lefebvre, S., Kwatra, V., Turk, G.: State of the Art in Example-based Texture Synthesis. In: Eurographics 2009, State of the Art Report, EG-STAR, pp. 93–117. Eurographics Association, Munich, Germany (2009)
41. Wei, L.Y., Levoy, M.: Fast texture synthesis using tree-structured vector quantization. In: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp. 479–488. ACM Press/Addison-Wesley Publishing Co. (2000)
42. Wexler, Y., Shechtman, E., Irani, M.: Space-time completion of video. IEEE Transactions on Pattern Analysis and Machine Intelligence **29**(3), 463–476 (2007). DOI 10.1109/TPAMI.2007.60. URL http://dx.doi.org/10.1109/TPAMI.2007.60
43. Zhou, Y., Shi, H., Lischinski, D., Gong, M., Kopf, J., Huang, H.: Analysis and controlled synthesis of inhomogeneous textures. Computer Graphics Forum **36**(2), 199–212 (2017)