Pacific Graphics 2015
N. J. Mitra, J. Stam, and K. Xu
(Guest Editors)

Volume 34 (2015), Number 7

Brushables: Example-based Edge-aware Directional Texture Painting

M. Lukáč,[1†] J. Fišer,[1] P. Asente,[2] J. Lu,[2] E. Shechtman,[2] and D. Sýkora[1]

[1]CTU in Prague, FEE, [2]Adobe Research

Figure 1: Examples of images synthesized using our method (right) generated from various sources (left). Our method simultaneously produces meaningful boundaries and interior structures, with textural features that respect the shape and direction specified by the user. *Source credits: denim: inxti @ shutterstock; plank: My Life Graphic @ shutterstock; grass: varuna @ shutterstock; cookie: Alessandro Paiva @ rgbstock*

Abstract

In this paper we present Brushables—a novel approach to example-based painting that respects user-specified shapes at the global level and preserves textural details of the source image at the local level. We formulate the synthesis as a joint optimization problem that simultaneously synthesizes the interior and the boundaries of the region, transferring relevant content from the source to meaningful locations in the target. We also provide an intuitive interface to control both local and global direction of textural details in the synthesized image. A key advantage of our approach is that it enables a "combing" metaphor in which the user can incrementally modify the target direction field to achieve the desired look. Based on this, we implement an interactive texture painting tool capable of handling more complex textures than ever before, and demonstrate its versatility on difficult inputs including vegetation, textiles, hair and painting media.

1. Introduction

Example-based image synthesis enables transfer of visual characteristics from a given exemplar to a user-defined target image [Ash01, HJO*01]. In this context a *texture-by-numbers* metaphor is typically used to guide the transfer of textural information between specific locations in the source and target images. Ritter et al. [RLC*06] showed that the quality of the synthesis can be improved when the algorithm takes into account specific effects that occur close to the boundaries of individual segments. This *edge-aware* approach was recently improved by Lukáč et

† e-mail:lukacmi1@fel.cvut.cz

al. [LFB*13] who showed how to synthesize boundary effects in a *direction-aware* manner independently from the interior—the direction of synthesized boundary features exactly follows the direction of the boundary shape. Direction awareness was also previously used in general texture synthesis [ZZV*03, LH06, ELS08, DBP*15] to allow control of the orientation of the synthesized texture.

A key limitation of existing techniques is that they treat edge and direction awareness independently, making it hard to produce images where the prescribed directionality of the shape's interior interacts with the appearance of its boundaries.

In this paper, we propose a novel method for interactive example-based image synthesis that combines edge and direction awareness in a single algorithm. While these features are useful independently to synthesize textural areas and linear edge structures with user-specified orientation, combining them enables complex shape-aware effects that no previous method can handle. See, e.g., examples in Figure 1 where the appearance of boundaries (e.g., blades of grass or hair ends) depends on a specific context that is given by the directionality of the interior.

Our method builds upon the popular patch-based optimization scheme originally developed by Wexler at al. [WSI07] and later extended by others [BSFG09, DSB*12, KNL*15]. A key contribution of our work is that we provide a new extension of the original Wexler et al. formulation that combines both direction and edge awareness into one optimization problem. We further improve the visual quality of the synthesized result using a novel coherence weighting mechanism.

We also propose a unified interactive framework that helps the user prepare the necessary input data for the synthesis. We extend previous related techniques for detecting [KLC09, Kyp11] and authoring [ZMT06, FSDH07] direction fields by creating a new *signed* direction field. The sign was not considered previously, and we show that it helps the user specify semantically meaningful configurations where unsigned orientation fields are insufficient (see, e.g., direction of hair/grass growth in Figure 1).

2. Related Work

One of the first instances of combining texture synthesis with a painting interface was Synthesizing Natural Textures [Ash01]. The user painted an output suggestion in the color domain, and the synthesis created output that roughly matched the colors. However, color information was not enough to finely specify texture areas.

Image Analogies [HJO*01] alleviated this limitation with a texture-by-numbers approach. The user pre-segmented the input image and directly painted a segmentation mask. However, the lack of additional information about boundary orientation led to visible inconsistencies.

Painting with Texture [RLC*06] represented a further development in this area. It was the first approach explicitly designed for synthesizing stroke interactions and texture edge effects by introducing a shape mask into the patch distance term. The mask provided rudimentary edge awareness, but its small size could not represent subtle orientation changes and larger sizes would make the synthesis over-constrained, causing visible repetitions and other artifacts.

Painting by Feature [LFB*13] presented an improvement over the previous techniques by treating lines and edges separately from the interior texture. Instead of relying on pre-segmented input images, the user interactively selected a line feature or a texture to be used as an example and then painted them into the output canvas. Despite full creative freedom, this technique could become tedious, requiring painstaking boundary tracing even when edges were obvious. This method also did not provide an explicit control over the directionality of the texture in the interior regions.

RealBrush [LBDF13] is a canonical example of stroke synthesis, capable of transferring the directionality and edge effects of the input strokes directly to the result using a painting metaphor. However, since it uses a lengthwise cut-and-stitch approach instead of full synthesis, it is strictly limited to 1D curves and cannot synthesize arbitrary area structures. Other stroke synthesis systems [LBW*14, ZLL13] typically suffer from the same limitation.

Accounting for directionality in texture synthesis is a proven idea. It can compensate for transformations [ELS08, LH06] or allow specification of direction in textures [ZZV*03, DBP*15]. Detecting orientation in images is also crucial for various stylization techniques [HE04, KLC09, Kyp11]. However, a painting scenario such as ours requires further considerations. The direction fields should be authored seamlessly using the basic brush metaphor, and the detection needs to be configurable to ensure compatibility of input and output direction fields.

Structure tensors [BWBM06] and edge tangent flow [KLC07] are common techniques to detect orientation in textures. Their key limitation is that they cannot provide consistent direction: the orientation sign is either omitted or inconsistent in the final solution. However, this is crucial in our scenario because real textures typically contain asymmetric structures. Although there are techniques that try to find consistent direction [KLC09, XCOJ*09], they typically fail on larger scales or when singularities are present in the input field.

User-guided authoring of vector fields has been extensively studied in the context of 3D surfaces [ZMT06, FSDH07, CDS10, MBS*11]. Although these techniques compute smoothly varying vector fields from a sparse set of user-provided constraints, their main drawback is that every new constraint has a global impact on the resulting field. In

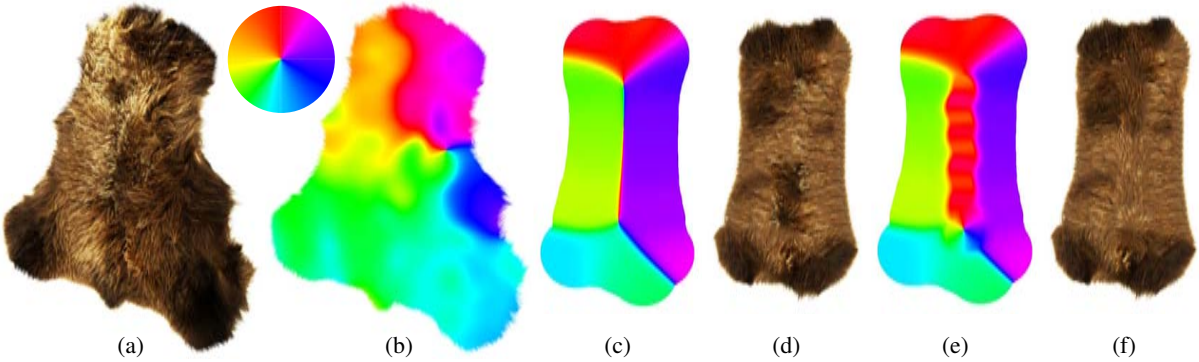


Figure 2: An example of Brushables workflow: (a) selected source image, (b) detected source direction field (direction wheel for reference), (c) hand-drawn stroke defining a target mask and direction field, (d) synthesis result, (e) refined target direction field, (f) refined result

our scenario we would like to modify the existing field on-the-fly by adding new directional strokes whose local impact is controlled by the user.

Optimization-based texture synthesis methods [KEBK05, WSI07] are the current state of art for synthesis applications [DSB*12, FLJ*14, KNL*15]. They accurately reproduce exemplar structures at interactive rates, thanks to fast approximate nearest-neighbor search [BSFG09]. We take advantage of the flexibility of this framework to introduce edge and direction awareness, and make adaptations to mitigate artifacts introduced by the free-form nature of our scenario.

For edge awareness, we build upon shape descriptors, commonly used in computer vision [BMP02, BM01]. They examine large areas of the shape to properly consider context, which is computationally expensive. Because texture synthesis requires numerous evaluations in a short timeframe, this can be a bottleneck.

3. Our Approach

Figure 2 illustrates the workflow of our method. The user starts with a regular RGB image and uses interactive image segmentation and matting to extract the area of interest along with the opacity values. The resulting RGBA image source S then serves as the basis for further processing (see Figure 2a).

Initially we take all pixels in S with non-zero alpha to form a binary *shape mask* M_s and then let the user determine the *edge extent*, i.e., how wide the boundary effects are. Finally, we employ *direction analysis* to obtain a source direction field d_s with a desired level of smoothness and consistent sign of the tangent vectors (see Section 3.1).

In the following painting phase, the user uses a brush tool to paint a mask that defines the set of pixels to be synthesized—the direction field d_t and its shape mask M_t. Then the use can alter or refine the target direction field (the *combing* process). For this our novel *direction diffusion* algorithm (Section 3.2) gives precise control over the stroke extent, seamlessly combining multiple strokes, and combining new strokes with the pre-existing direction field.

Finally, given the source image S, source and target masks (M_s and M_t) and direction fields (d_s and d_t) we run our direction- and edge-aware texture synthesis (Section 3.3). We synthesize the output texture, using a novel *Shape Hint* to ensure that boundary effects are synthesized appropriately in a context-sensitive way, enforcing the prescribed direction, and using a *coherence weighting* mechanism to improve the final visual quality of the synthesized image T even under strong non-rigid deformation.

3.1. Direction Analysis

Before we can paint taking the directionality of the source into account, we need to estimate it. Our first step is to create a direction field d_s that specifies local direction at all pixels of S. To support arbitrary input exemplars and have a self-contained approach, we determine the direction field using only the RGB color information.

For best results, a reasonable direction field d_s should be locally smooth and perpendicular to the gradient field of S—a tangent field. Because smoothness and perpendicularity cannot usually be satisfied simultaneously, additional filtering is required. We also take the sign of the tangent vectors into account, since they are often semantically significant.

Estimation of smooth tangent fields has been explored before, predominantly in image stylization techniques [KLC07, Kyp11]. However, these approaches typically ignore the sign of the tangent vector, since the filters they ultimately employ are symmetric with respect to the sign. In particular, the multi-lateral filter employed by Kang et al. uses a non-linear term to preserve the sign of the tan-

gent vectors. In such case, flipping the signs of some of the tangents in the initialization phase will not affect the magnitudes or absolute direction of the tangents in the resulting tangent field, merely their signs. This means that we can solve sign harmonization independently as a pre-processing pass and then apply one of the filters to get a coherent, smooth result.

We base our sign harmonization on the *edge tangent flow* (ETF) filter [KLC07], but we improve the initialization. We fix the sign of the pixel with the greatest gradient magnitude and use a breadth-first propagation to harmonize the tangents along with their signs. Figure 3 shows how this method eliminates the 180-degree discontinuities present in earlier methods.

3.2. Direction Diffusion

In previous approaches [DSB*12, LFB*13], the texture direction emerges implicitly from the color domain so as to match the boundary conditions. In contrast, we give the user explicit control over texture direction, much like stroke synthesis approaches do [ZLL13, LBW*14]. Given a region painted by the user with a variable-width brush, we determine the direction field d_t, assigning a direction to every pixel in the region.

Like stroke synthesis, we are given a user-specified 1D stroke path with an instantaneous direction at every path sample. We propagate the sparse direction samples to the entire stroke area. As an act of painting, the effect of brushing should be local, with its influence strictly limited to the area within the brush footprint leaving the rest of the image unaffected. To avoid synthesis artifacts, we also must ensure that we do not create discontinuities in the direction field at the brush boundary and that the target direction field has a similar level of smoothness to the direction field of the input.

Related approaches use various optimization processes to construct a smooth direction field from sparse user-specified constraints [ZMT06, FSDH07]. However, these techniques are global by nature and do not provide for a localized, controlled way to combine new strokes with an existing direction field, which is needed to permit *combing* and general refinement. We use a kernel-based diffusion scheme to smoothly diffuse and blend the direction of an arbitrary number of strokes of variable radius, while also permitting blending with a pre-existing field.

Given a stroke path K consisting of all points $k \in K$, we calculate the direction $d_k(p)$ diffused from this stroke at a point p as follows:

$$d_k(p) = \frac{1}{w_k(p)} \int_{k \in K} G(||p \quad k||^2, \sigma_k^2) \cdot d'(k) \qquad (1)$$

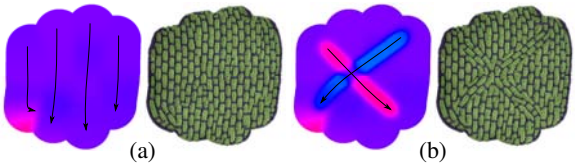
where $G(x, \sigma_k^2)$ is a gaussian kernel with the standard deviation set to half the stroke width, $d'(k)$ is the local normalized tangent, and

$$w_k(p) = \int_{k \in K} G(||p \quad k||^2, \sigma_k^2) \qquad (2)$$

This yields a smooth interpolation that can be evaluated analytically if the input stroke is approximated as a polyline, and the generalization to multiple simultaneous strokes is straightforward (see Figure 4a). If we need to combine the diffused direction of the current stroke with the aggregated direction field of all the previous strokes (as in Figure 4b), we calculate the convex mix of the previous value $d^{n\ \ 1}(p)$ with the new one $d_k(p)$ like so:

$$d^n(p) = w_s(p) \cdot d_s(p) + (1 \quad w_s(p)) \cdot d^{n\ \ 1}(p) \qquad (3)$$

assuming $w_s(p)$ is clamped to remain in the convex interval $\langle 0, 1 \rangle$.



(a) (b)

Figure 4: A demonstration of direction field authoring and refinement. (a) a composition of thick strokes made with a 120px wide brush next to its synthesis result; (b) direction field with two 80px refinement strokes and the refined synthesis result.

3.3. Example-based Synthesis

Once source and target direction fields d_s and d_t are prepared we proceed to the synthesis phase, generating the output image while respecting the principles of edge and direction awareness we have described earlier.

We build our synthesis algorithm upon established the patch-based optimization framework introduced originally by Wexler et al. [WSI07]. We chose this framework for its flexibility: we can substantially alter its behavior by substituting our own patch distance measure and patch voting logic, making it fit our own requirements.

We introduce edge-awareness into the synthesis by adding a new *shape distance term* to the energy function we minimize:

$$E(T,S) = \sum_{q \in T} \min_{p \in S} \left(D_{\text{patch}}(p,q) + \lambda D_{\text{shape}}(p,q) \right) \qquad (4)$$

$D_{\text{patch}}(p,q)$ measures the color distance of patches and $D_{\text{shape}}(p,q)$ the distance of local shapes around pixels $p \in S$ and $q \in T$.

Direction awareness is added to these distance measures by taking local direction at both p and q into account. We do this by introducing a rotation operator $\oslash\alpha_{pq}$, which rotates the local frame of reference for the patch or shape descriptor

(a) (b) (c) (d) (e)

Figure 3: An illustration of the output of Kang et al.'s [KLC07] method for orientation detection with and without our unified tangent sign initialization: (a) original synthetic image (radial stripes), (b) false-colour visualization of the initial direction field (gradients rotated 90°to the left), (c) converged result after a few ETF filter iterations, (d) initial direction field after our unified tangent sign initialization, (e) result after single ETF filter iteration applied on our unified tangent sign initialization.

by the difference in local direction at p and q, i.e., $\alpha_{pq} = d_t(q) - d_s(p)$.

We can then calculate the color distance as the direction-aware *sum of squared differences*:

$$D_{\text{patch}}(p,q) = \left|\left|\mathbf{P}^s_p - \mathbf{P}^t_q \oslash \alpha_{pq}\right|\right|^2 \tag{5}$$

between the source patch $\mathbf{P}^s_p$ centered on $p \in S$ and the rotated target patch $\mathbf{P}^t_q$ centered on $q \in T$. Similarly, the direction-aware shape distance is evaluated as:

$$D_{\text{shape}}(p,q) = \chi^2\left(\mathbf{H}^s_p, \mathbf{H}^t_q \oslash \alpha_{pq}\right) \tag{6}$$

i.e., the distance between source and target *Shape Hint* histograms described below, which introduce shape awareness by considering both the spatial distance from the texture boundary, and its shape relative to the local direction field.

Figure 5: Importance of Shape Hints: (a) synthesis without Shape Hint (using just distance to the boundary) and (b) synthesis with Shape Hint.

Shape Hint: To introduce edge-awareness into the synthesis we use *Shape Hints*—a local shape descriptor derived from the *shape context* [BMP02], which we have simplified and adapted for interactive use. Shape descriptors like these are a powerful tool commonly used to find similar locations within shapes. Compared to previous context aware solutions based on a distance transform [LH06, BCK*13], a shape descriptor considers a larger context, allowing it to distinguish between locations at edges and corners or around interior holes; this is vital for our concept of edge-awareness, since it lets us pick patches from appropriate regions more contextually (see Figure 5). It is also more flexible than comparing mask patches, as in Painting with Texture [RLC*06]; the distance measure is continuous rather than discrete and degenerates gracefully, without overconstraining the synthesis at texture edges.

Like the shape context, our descriptor counts the edge pixels that fall into "bins" mapped to image space. These counts are then treated as histograms of edge pixels and can be compared using the χ^2 metric. This creates a descriptor that is capable of capturing the shape of the object boundary with a configurable level of tolerance to high-frequency variations, based on how large is the spatial support of the bins. To date, performance considerations precluded the use of shape context in texture synthesis, as typically local descriptors need to be evaluated repeatedly at many points of the image, and the computational complexity of evaluating a shape context scales quadratically with its radius.

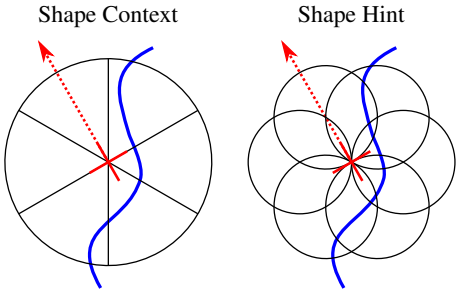


Figure 6: Shape and arrangement of bins in an oriented single-layer *shape context* (left) and in our *Shape Hint* (right).

To overcome this limitation, we propose an adaptation wherein we change the shape of the bins used to count edge pixels (see Figure 6). Instead of annular sections, we use circular bins, similar to image descriptors like

FREAK [AOV12], but we keep the shape context's compact representation based on edge pixel counting and its method of calculating similarity.

As circles are rotationally invariant, the shape of the bin becomes constant with respect to both the orientation of the descriptor as a whole and the bin's position therein. The value of any bin at any point can thus be pre-computed by convolving the edge pixel map with a disc filter of the appropriate radius, and consequentially, we can evaluate the descriptor with a constant number of bitmap queries regardless of its spatial support or the number of edge pixels in the image.

This not only leads to faster evaluation, making use in texture synthesis possible, but also permits free-form continuous rotations of the descriptor at no additional computational cost. We have found it sufficient to only use a single radial layer of bins, although the descriptor naturally generalizes to multiple layers.

In synthesis, we use the source and target masks M_T and M_S to calculate the Shape Hints, with the radius of the descriptor set to the *edge extent* that the user defined earlier in the source analysis phase. This value should roughly correspond to the width of the boundary effects, i.e., how "deep" into the texture they extend. Content within this range is implicitly treated as the boundary, while content deeper inside is considered to be in the interior.

Alpha Channel: To further improve the quality of the synthesis at boundaries we add an alpha mask as an additional pixel channel. This has two effects. It lets us synthesize opacity, and together with the Shape Hint, guides the synthesis towards a solution where pixels close to boundaries in the source are more likely to be matched with boundary pixels in the target. To give the opacity comparable weight to color we multiply the difference in alpha channel by 3 when computing the sum of squared differences in (5).

While the alpha channel gives us the ability to synthesize opacity and "fading out" at the boundaries, it is in itself not sufficient to capture longer-range edge effects, and cannot discriminate boundaries from the interior in textures with partially transparent interiors. Therefore, a combination of alpha channel synthesis and shape matching is optimal for synthesis of edge effects in our scenario.

Optimization: To minimize (4) we use the Expectation-Maximization optimization outlined by Wexler et al [WSI07] that consists of alternating *search* and *voting* steps on an image pyramid in a coarse-to-fine order. To improve texture coherence and richness in the synthesized image, we propose an improvement to the voting step to take both local nearest-neighbor field coherency and the color histograms of both images into account. When evaluating the final color $C(p)$ of a pixel p, we iterate through the overlapping patches mapped to its neighborhood and perform a weighted average of the candidates c_x gathered from them:

$$C(p) = \frac{\sum_{q \in \mathcal{N}_p} w_c(q) \cdot w_h(q) \cdot C(q)}{\sum_{q \in \mathcal{N}_p} w_c(q) \cdot w_h(q)} \tag{7}$$

where w_h is the *color histogram weight* of the candidate pixel, as detailed by Kopf et al [KFCO*07] and w_c is the *coherence weight*, which serves to propagate coherent arrangements of patches from the source. As described in the original paper, the histogram weight promotes pixel candidates with relatively underrepresented colors, improving the diversity of the synthesized image.

The coherence weight is vital in our scenario, since free-form rotations of the texture tend to induce non-rigid mapping in the nearest-neighbor field, which in turn causes blurry and visually displeasing results (see comparison in Figure 7). By increasing the weight of coherently-mapped configurations of patches, we encourage forming larger, coherently mapped areas over multiple iterations. This preserves high-frequency detail and causes less blurring.

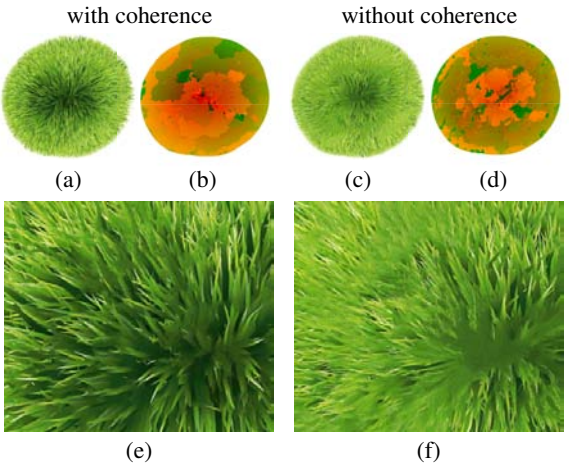


Figure 7: A comparison of results (a,c) and corresponding nearest-neighbor fields (b,d) synthesised with (left) and without (right) the coherence weight. The details (e,f) show how structural details of individual blades are better preserved with the coherence weight. Note also how the coherence weight leads to larger patches in the nearest-neighbor fields.

To calculate the coherency weight, we examine the coherence of mapped pixel configurations as follows:

In effect, a nearest-neighbor match is a rigid mapping from T to S. The matched coordinates and relative rotation at a pixel q thus define a mapping $\mathbf{R}_q$, which maps the pixel grid in T to a rotated and offset pixel grid in S. Because the optimization is based on the assumption that these mappings are approximately identical for the group of pixels within the area of a patch, we design our coherency measure as a quantification of how this assumption holds. To evaluate this

measure, we examine the patch neighborhood of a pixel q_0 and the induced mappings therein (c.f. Figure 8):

$$w_c(q_0) = \sum_{q \in \mathcal{N}_{q_0}} G(||\mathbf{R}_{q_0}(q) \quad \mathbf{R}_q(q)||^2, \sigma_c^2) \qquad (8)$$

where σ_c^2 is the *coherency range*, which we set to 2 throughout.

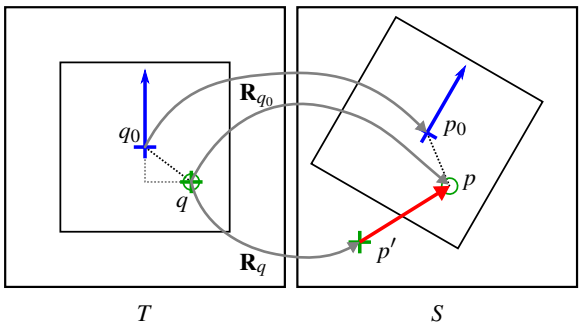


Figure 8: Calculating the coherence of a patch: we examine all pixels q in a patch around the pixel q_0 in the target image T. The position of pixel q is projected into the pixels p and p' in the source image S using both the rigid transformation induced by the match at q_0: $p = \mathbf{R}_{q_0}(q)$ and its own transformation: $p' = \mathbf{R}_q(q)$. The more coherent the matching is, the lower the sum of distances $||p \quad p'||$ (red arrow).

Multiplying these weights, along with the guaranteed range on both of them, ensures that the weighting scheme degenerates gracefully in any edge case.

3.4. Implementation Details

We have implemented the described algorithm in C++11 and run it on a desktop computer. The synthesis takes approximately 5 seconds for a megapixel output image, with 80% of the time spent calculating the nearest-neighbor field. The texture analysis step was more computationally intensive, taking up to 30 seconds for larger settings of the ETF filter; however, this only needs to be done once for each source texture as a pre-process, and the results can be efficiently stored. The rest of the method operates interactively.

As most of our parameters have intuitive semantics, they were set contextually as appropriate. The range of the edge tangent flow filter was usually set to a default value of 10 pixels. This setting was only increased for noisier textures to approximately 35 pixels. λ in Equation 4 was hardcoded to a value of 25 (equal to the number of pixels in a patch used to measure color distance), and the width of the brush was interactively adjusted as appropriate.

Edge extent was the crucial parameter to achieve edge awareness; setting it too low can cause boundary patches to be randomly used in the interior, while setting it too high can cause the extent of the synthesized texture to visibly deviate inwards from the user-specified mask. Ultimately, the value in pixels should be set approximately equal to the width of the boundary effects the user wishes to capture.

4. Results

Figures 1 and 14 shows the synthesis results of a variety of natural textures. Our approach coherently synthesizes textured outputs with direction configurations not present in the original source (see e.g., the crochet results). Plank example demonstrates how our algorithm picks the semantically correct edge features according to local direction. The grass and the colored pencil examples show that even transparency is synthesized correctly both on the boundary and in the interior. The braided wig example serves to highlight the strength of shape descriptor-based edge awareness; the narrower sections are synthesized from braid patches, while the wider parts are synthesized out of the upper, combed part of the exemplar. Again, current approaches do not have such capabilities. The ornamental leaves are an example of a relatively simpler stroke synthesis application. It shows how our approach organically synthesizes branching by virtue of not relying on stroke semantics. The red wig shows how locally-variant anisotropic textures can be coherently deformed to novel configurations.

Note that after texture analysis, the only user input we require are the brush strokes. Thus, our tool places no more burden on the end user than a regular brush. The overall interaction takes only a couple of seconds, depending mainly on the ability of the user to draw individual strokes (see supplementary video for an example of interactive sessions). This brings an improvement over Painting by Feature [LFB*13], which requires more elaborate input to achieve similar results (see Figure 10).

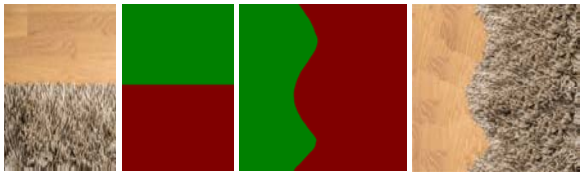


Figure 9: A example result when using multiple textures. The top row is the example and its segmentation, bottom row the synthesized output and its painted segmentation. *Source credit: Radu Bercan @ shutterstock*

Our method can also be easily extended to process images with multiple, segmentable textures (see Figures 9 and 11). In this case, we require that a segmentation map be provided for the input texture, and the direction brush is concurrently used to paint also the output segmentation. The synthesis is then adjusted so that it only maps patches between compatible segments. Segment boundaries are considered in the same way as foreground boundaries for the Shape Hint.

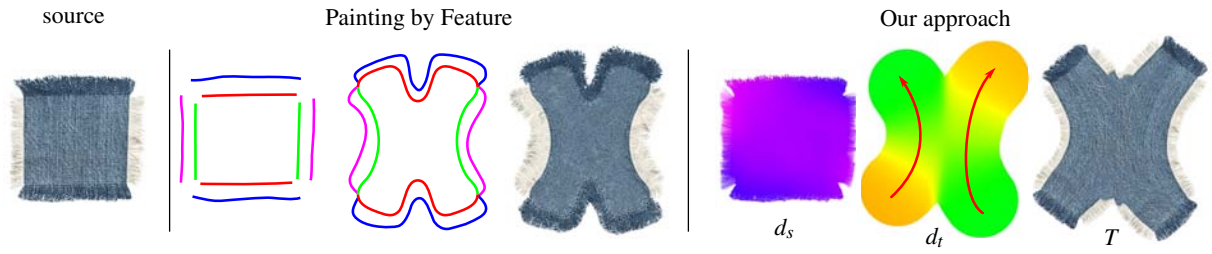


Figure 10: Comparison of the amount of user interaction required to create a similar output using Painting by Feature (PBF) [LFB*13] and our approach. Each colored line in the PBF example represents a user stroke (area selections are not shown); in contrast, our approach produced the result with only two strokes. Also compare the coherence of texture on the interior and the tassels.

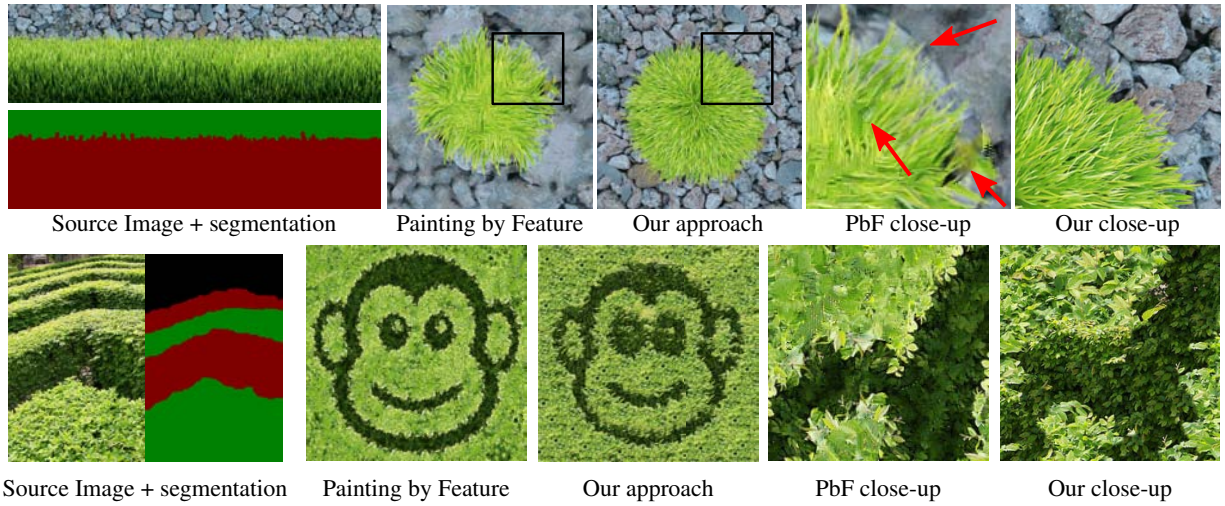


Figure 11: Comparison of our approach with Painting by Feature (PBF) [LFB*13] with respect to ability to handle edges with highly varying width. Note how our unified approach integrates interiors with edges smoothly, whereas in the PBF output there are discontinuities between the areas synthesized as edges and those synthesized as interiors (see red arrows). In the bottom example, PBF is able to more closely match the user-specified shape, but does so at the cost of faithfulness to the example and visual richness. *Source credit: monkey: © ACM; hedge source: Joe Shlabotnik @ flickr*

Multiple texture extension allows us to make a comparison with Painting by Feature [LFB*13] (see Figure 11 and supplementary material). Our method produces comparable or better visual quality without the necessity to use a custom synthesis algorithm for the boundaries. It also notably improves the look of the interior parts by maintaining the appearance of the original source and creating seamless transitions from the edge that follows prescribed direction field.

Our method can also be used to synthesize example-based brush strokes of comparable quality to those produced by RealBrush framework [LBDF13] (see Figure 12). In addition, the same algorithm can be applied to fill larger areas, which the original RealBrush method cannot do.

In Figure 13 we show results where only the edge or direction awareness is taken into account. This example demonstrates limitations of previous approaches (such as [RLC*06] or [LH06]) where a joint edge- and direction-aware formulation was not considered.

4.1. Limitations

Our algorithm does not automatically take changes in texture scale into account, nor does it natively compensate for perspective. Support for these could be added by pre-processing the input image to compensate for these.

Because our algorithm does not take advantage of any domain knowledge, it cannot replicate stroke-specific effects that require such knowledge. Most significantly, the smudging and smearing effects supported by RealBrush [LBDF13] cannot be replicated. Instead, overlapping strokes merge into a single larger area and are synthesized as such (see Figure 11 right).

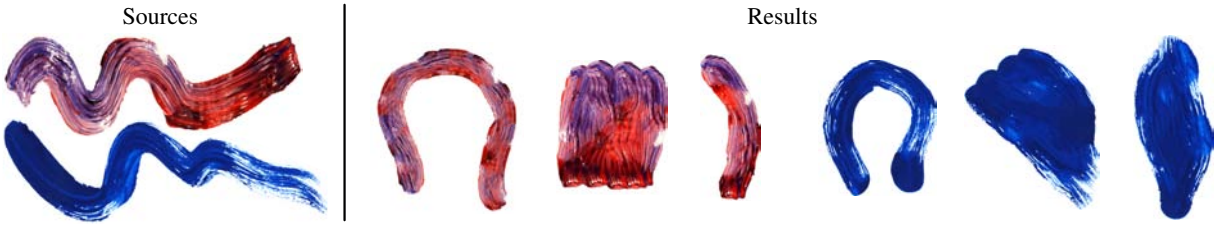
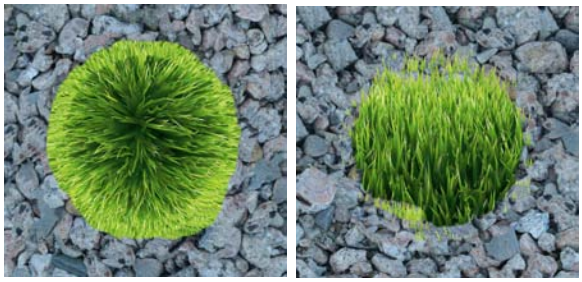


Figure 12: Brushables can also be used in the RealBrush scenario [LBDF13]. Our approach can synthesize new strokes like RealBrush can, and also synthesize regions of arbitrary shape.

Furthermore, because we rely on an area representation rather than an outline-based one, our approach does not natively handle interior lines like Painting by Feature [LFB*13] does. This effect could be emulated by selecting the line in the example as a separate texture, painting that and combining the results. Still, the nature of our brush-based interaction model makes this less convenient than similar operations are for vector-based tools.

In some textures, there may be hidden variables not related to direction that affect incidence of features both on the edges and on the interior; these might include e.g. the holes in the cracker, or the precise position of the hairband in the braid. In such cases, our approach is unable to distinguish the underlying semantics and will distribute these features randomly. The ability to specify manual constrains, such as the ones used in appearance-space texture synthesis [LH06], could allow the user to resolve these cases.



Only direction-aware Only edge-aware

Figure 13: Results from Figure 11 with the shape awareness and direction awareness turned off. Those examples demonstrate importance of joint formulation proposed in our framework and illustrate limitation of previous approaches, which take into account only direction [LH06] or edge [RLC*06] awareness.

Our algorithm also exhibits some of the artifacts of the original synthesis method of Wexler et al. [WSI07], namely the repetition of textural features. Extensions to this optimization scheme that eliminate these have been proposed [KNL*15, JFA*15]; we consider these to be orthogonal to, and compatible with, our work.

5. Conclusion and Future Work

As discussed above, our approach handles complicated natural textures using a simple mode of interaction demonstrated earlier. Adding direction awareness to the synthesis process lets us handle textures with locally-variant anisotropic properties without requiring large exemplars or losing information. Our direction detection and authoring framework give users control over the output direction field that is semantically significant for many textures.

Adding the shape hint to texture synthesis enables robust handling of edge effects. Combined with alpha-channel synthesis, our approach can reproduce edge effects present in partially transparent textures. As a result, edges need not be explicitly drawn by the artists any more.

When combined, these two features become even more powerful, allowing semantically significant edge areas to be used for synthesis in different places. This allows artists to use previously unaccessible textures for true interactive texture painting.

For future work, we would like to better handle the cases where the direction configurations in the source do not match the target direction field. One possible solution is to automatically adapt the target direction field in a constrained and meaningful way. Furthermore, we would like to experiment with our shape hint in the domain of shape synthesis. It might be able to give rough user sketches the same type of high-level detail that a source shape does. Another possible avenue is to synthesize the mixing of textures using a blending approach like Image Melding [DSB*12].

Our approach integrates naturally into digital painting pipelines, thanks to its intuitive mode of interaction. Its ability to handle painting media exemplars lends itself to the creation of digital art. The ability to synthesize complex natural textures with edge effects make it useful for photo editing or matte painting applications.

Acknowledgements

We would like to thank all anonymous reviewers for their constructive comments. This research began as an internship by Michal Lukáč at Adobe Research and has been

Figure 14: Various sources (top): cracker, crochet, denim, sample of color pencil, bread, red wig, braided wig, ornamental leaves, plank, and grass were used to synthesize target images (below). Note how our approach handles both linear structures and regions with boundaries and how user-specified directions are gracefully preserved in the result. *Source credits: cookie: Alessandro Paiva @ rgbstock; crochet: anneheathen @ flickr; denim: inxti @ shutterstock; bread: Giles Hodges @ DeviantArt; red wig: Lenor Ko @ shutterstock; braided wig: Karina Bakalyan @ shutterstock; ivy leaves: Michael & Christa Richert @ rgbstock; plank: My Life Graphic @ shutterstock; grass: varuna @ shutterstock*

supported by the Technology Agency of the Czech Republic under research program TE01020415 (V3C) and by the Grant Agency of the Czech Technical University in Prague, grant No. SGS13/214/OHK3/3T/13 (Research of Progressive Computer Graphics Methods).

References

[AOV12] ALAHI A., ORTIZ R., VANDERGHEYNST P.: Freak: Fast retina keypoint. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (June 2012), pp. 510–517. 6

[Ash01] ASHIKHMIN M.: Synthesizing natural textures. In *Proceedings of Symposium on Interactive 3D graphics* (2001), pp. 217–226. 1, 2

[BCK*13] BÉNARD P., COLE F., KASS M., MORDATCH I., HEGARTY J., SENN M. S., FLEISCHER K., PESARE D., BREEDEN K.: Stylizing animation by example. *ACM Transactions on Graphics 32*, 4 (2013), 119. 5

[BM01] BERG A. C., MALIK J.: Geometric blur for template matching. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (2001), vol. 1, pp. 607–614. 3

[BMP02] BELONGIE S., MALIK J., PUZICHA J.: Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 4 (2002), 509–522. 3, 5

[BSFG09] BARNES C., SHECHTMAN E., FINKELSTEIN A., GOLDMAN D. B.: PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics 28*, 3 (2009), 24. 2, 3

[BWBM06] BROX T., WEICKERT J., BURGETH B., MRÁZEK P.: Nonlinear structure tensors. *Image and Vision Computing 24*, 1 (2006), 41–55. 2

[CDS10] CRANE K., DESBRUN M., SCHRÖDER P.: Trivial connections on discrete surfaces. *Computer Graphics Forum 29*, 5 (2010), 1525–1533. 2

[DBP*15] DIAMANTI O., BARNES C., PARIS S., SHECHTMAN E., SORKINE-HORNUNG O.: Synthesis of complex image appearance from limited exemplars. *ACM Transactions on Graphics 34*, 2 (2015). 2

[DSB*12] DARABI S., SHECHTMAN E., BARNES C., GOLDMAN D. B., SEN P.: Image Melding: Combining inconsistent images using patch-based synthesis. *ACM Transactions on Graphics 31*, 4 (2012), 82. 2, 3, 4, 9

[ELS08] EISENACHER C., LEFEBVRE S., STAMMINGER M.: Texture synthesis from photographs. *Computer Graphics Forum 27*, 2 (2008), 419–428. 2

[FLJ*14] FIŠER J., LUKÁČ M., JAMRIŠKA O., ČADÍK M., GINGOLD Y., ASENTE P., SÝKORA D.: Color Me Noisy: Example-based rendering of hand-colored animations with temporal noise control. *Computer Graphics Forum 33*, 4 (2014), 1–10. 3

[FSDH07] FISHER M., SCHRÖDER P., DESBRUN M., HOPPE H.: Design of tangent vector fields. *ACM Transactions on Graphics 26*, 3 (2007), 56. 2, 4

[HE04] HAYS J., ESSA I.: Image and video based painterly animation. In *Proceedings of International Symposium on Non-photorealistic Animation and Rendering* (2004), pp. 113–120. 2

[HJO*01] HERTZMANN A., JACOBS C. E., OLIVER N., CURLESS B., SALESIN D. H.: Image analogies. In *Proceedings of SIGGRAPH 2001* (2001), pp. 327–340. 1, 2

[JFA*15] JAMRIŠKA O., FIŠER J., ASENTE P., LU J., SHECHTMAN E., SÝKORA D.: Lazyfluids: Appearance transfer for fluid animations. *ACM Transactions on Graphics 34*, 4 (2015). 9

[KEBK05] KWATRA V., ESSA I. A., BOBICK A. F., KWATRA N.: Texture optimization for example-based synthesis. *ACM Transactions on Graphics 24*, 3 (2005), 795–802. 3

[KFCO*07] KOPF J., FU C.-W., COHEN-OR D., DEUSSEN O., LISCHINSKI D., WONG T.-T.: Solid texture synthesis from 2D exemplars. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 26*, 3 (2007), 2. 6

[KLC07] KANG H., LEE S., CHUI C. K.: Coherent line drawing. In *Proceedings of International Symposium on Non-Photorealistic Animation and Rendering* (2007), pp. 43–50. 2, 3, 4, 5

[KLC09] KANG H., LEE S., CHUI C. K.: Flow-based image abstraction. *IEEE Transactions on Visualization and Computer Graphics 15*, 1 (2009), 62–76. 2

[KNL*15] KASPAR A., NEUBERT B., LISCHINSKI D., PAULY M., KOPF J.: Self tuning texture optimization. *Computer Graphics Forum 34*, 2 (2015), 349–360. 2, 3, 9

[Kyp11] KYPRIANIDIS J. E.: Image and video abstraction by multi-scale anisotropic Kuwahara filtering. In *Proceedings of International Symposium on Non-Photorealistic Animation and Rendering* (2011), pp. 55–64. 2, 3

[LBDF13] LU J., BARNES C., DIVERDI S., FINKELSTEIN A.: RealBrush: Painting with examples of physical media. *ACM Transactions on Graphics (Proc. SIGGRAPH)* (2013). 2, 8, 9

[LBW*14] LU J., BARNES C., WAN C., ASENTE P., MECH R., FINKELSTEIN A.: DecoBrush: Drawing structured decorative patterns by example. In *ACM Transactions on Graphics (Proc. SIGGRAPH)* (2014). 2, 4

[LFB*13] LUKÁČ M., FIŠER J., BAZIN J.-C., JAMRIŠKA O., SORKINE-HORNUNG A., SÝKORA D.: Painting by Feature: Texture boundaries for example-based image creation. *ACM Transaction on Graphics 32*, 4 (2013). 2, 4, 7, 8, 9

[LH06] LEFEBVRE S., HOPPE H.: Appearance-space texture synthesis. *ACM Transactions on Graphics 25*, 3 (2006), 541–548. 2, 5, 8, 9

[MBS*11] MAHARIK R., BESSMELTSEV M., SHEFFER A., SHAMIR A., CARR N.: Digital micrography. *ACM Transactions on Graphics 30*, 4 (2011), 100. 2

[RLC*06] RITTER L., LI W., CURLESS B., AGRAWALA M., SALESIN D.: Painting with texture. In *Proceedings of Eurographics Symposium on Rendering* (2006), pp. 371–376. 1, 2, 5, 8, 9

[WSI07] WEXLER Y., SHECHTMAN E., IRANI M.: Space-time completion of video. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*, 3 (2007), 463–476. 2, 3, 4, 6, 9

[XCOJ*09] XU K., COHEN-OR D., JU T., LIU L., ZHANG H., ZHOU S., XIONG Y.: Feature-aligned shape texturing. *ACM Transactions on Graphics 28*, 5 (2009), 108. 2

[ZLL13] ZHOU S., LASRAM A., LEFEBVRE S.: By–example synthesis of curvilinear structured patterns. *Computer Graphics Forum 32* (2013), 355–360. 2, 4

[ZMT06] ZHANG E., MISCHAIKOW K., TURK G.: Vector field design on surfaces. *ACM Transactions on Graphics 25*, 4 (2006), 1294–1326. 2, 4

[ZZV*03] ZHANG J., ZHOU K., VELHO L., GUO B., SHUM H.-Y.: Synthesis of progressively-variant textures on arbitrary surfaces. *ACM Transactions on Graphics 22*, 3 (2003), 295–302. 2