Computational Visual Media
DOI 10.1007/s41095-xxx-xxxx-x Vol. x, No. x, month year, xx–xx

Article Type (Research/Review)

A survey of the state-of-the-art in patch-based synthesis

Connelly Barnes[1]✉, **and Fang-Lue Zhang**[2]

Abstract This paper surveys the state-of-the-art of research in patch-based synthesis. Patch-based methods synthesize output images by copying small regions from exemplar imagery. This line of research originated from an area called "texture synthesis," which focused on creating regular or semi-regular textures from small exemplars. However, more recently, much research has focused on synthesis of larger and more diverse imagery, such as photos, photo collections, videos, and light fields. Additionally, recent research has focused on customizing the synthesis process for particular problem domains, such as synthesizing artistic or decorative brushes, synthesis of rich materials, and synthesis for 3D fabrication. This report investigates recent papers that follow these themes, with a particular emphasis on papers published since 2009, when the last survey in this area was published. This survey can serve as a tutorial for readers who are not yet familiar with these topics, as well as provide comparisons between these papers, and highlight some open problems in this area.

Keywords Texture, patch, image synthesis.

1 Introduction

Due to the widespread adoption of digital photography and social media, digital images have enormous richness and variety. Photographers frequently have personal photo collections of thousands of images, and cameras can be used to easily capture high-definition video, stereo images, range images, and high-resolution material samples. This deluge of image data has spurred research into algorithms that automatically remix or modify existing imagery based on high-level user goals.

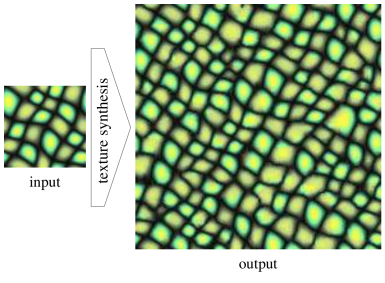


Fig. 1 An illustration of texture synthesis. A texture synthesis algorithm is given as input a small exemplar consisting of a regular, semi-regular, or stochastic "texture" image. The algorithm then synthesizes a large, seamless output texture based on the input exemplar. Reproduced with permission from Wei et al. [104], © The Eurographics Association 2009.

One successful research thread for manipulating imagery based on user goals is patch-based synthesis. Patch-based synthesis involves a user providing one or more exemplar images to an algorithm, which is then able to automatically synthesize new output images by mixing and matching small compact regions called *patches* or *neighborhoods* from the exemplar images. Patches are frequently of fixed size, e.g. 8x8 squares.

The area of patch-based synthesis traces its intellectual origins to an area called "texture synthesis," which focused on creating regular or semi-regular textures from small examples. See Figure 1 for an example of texture synthesis. A comprehensive survey of texture synthesis methods up to the year 2009 is available [104]. Since then, research has focused increasingly on synthesis of larger and more diverse imagery, such as photos, photo collections, videos, and light fields. Additionally, recent research has focused on customizing the synthesis process for particular problem domains, such as synthesizing artistic or decorative brushes, synthesis of rich materials, and synthesis for 3D fabrication.

1 University of Virginia, Charlottesville, VA 22904, USA. E-mail: connelly (at) cs.virginia.edu.

2 TNList, Tsinghua University, Beijing, P.R. China, P.C.: 100084. E-mail: z.fanglue (at) gmail.com.

Manuscript received: 2016-XX-YY; accepted: 2016-XX-YY.

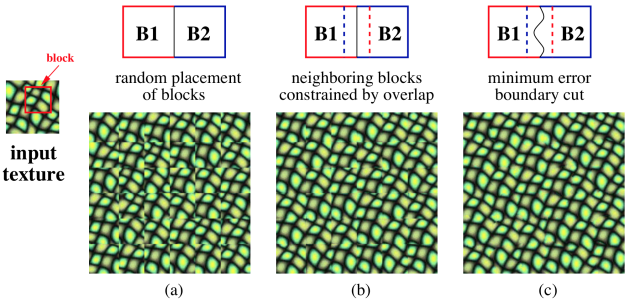


Fig. 2 An illustration of the patch-based synthesis method of image quilting [36]. At left is an input exemplar. At right, in (a), (b), and (c), output imagery is synthesized. In (a) and (b) are shown what we call the *matching* stage, where patches are selected according to different criteria: either random sampling (a) or based on a patch similarity term with previously selected patches (b). In (c) is shown what we call the *blending* stage, which composites and blends together overlapping patches. In this method, the "blending" is actually done based on a minimal error boundary cut. Reproduced with permission from Efros and Freeman [36], © Association for Computing Machinery 2001.

In this survey, we cover recent papers that follow these themes, with a particular emphasis on papers published since 2009. This survey also provides a gentle introduction to the state-of-the-art in this area, so that readers unfamiliar with this area can learn about these topics. We additionally provide comparisons between state-of-the-art papers and highlight open problems.

2 Overview

This survey paper is structured as follows. In Section 3, we provide an gentle introduction to patch-based synthesis by reviewing how patch-based synthesis methods work. This introduction includes a discussion of the two main stages of most patch-based synthesis methods: *matching*, which finds suitable patches to copy from exemplars, and *blending*, which composites and blends patches together on the output image. Because the matching stage tends to be inefficient, in Section 4, we next go into greater depth on accelerations to the patch matching stage. For the remainder of the paper, we then investigate different applications of patch-based synthesis. In Section 5-Section 7, we discuss applications to image inpainting, synthesis of whole images, image collections, and video. In Section 8, we then investigate synthesis algorithms that are tailored towards specialized problem domains, including mimicking artistic style, synthesis of artistic brushes and decorative patterns, synthesis for 3D and 3D fabrication, fluid synthesis, and synthesis of rich materials. Finally, we wrap up with discussion and a possible area for future work in Section 9.

3 Introduction to patch-based synthesis

There are two main approaches for example-based synthesis: pixel-based methods, which synthesize by copying one pixel at a time from an exemplar to an output image, and patch-based methods, which synthesize by copying entire patches from the exemplar. Pixel-based methods are discussed in detail by Wei et al. [104]. Patch-based methods have been in widespread use recently, so we focus on them.

We now provide a review of how patch-based synthesis methods work. In Figure 2, an illustration is shown of the patch-based method of image quilting [36]. There are two main stages of most patch-based synthesis methods: *matching* and *blending*.

The *matching* stage locates suitable patches to copy from exemplars, by establishing a correspondence between locations in the output image being synthesized, and the input exemplar image. In image quilting [36], this is done by laying down patches in raster scan order, and then selecting out of a number of candidate patches the one that has the best agreement with already placed patches. This matching process is straightforward for small textures but becomes more challenging for large photographs or photo collections, so we discuss in more depth different matching algorithms in Section 4.

Subsequently, the *blending* stage composites and blends patches together on the output image. See Figure 3 for an illustration of the different patch blending methods discussed here. For sparse patches that have a relatively small overlap region, blending can be done by simply compositing irregularly-shaped patches [90], using a blending operation in the overlap region [70], or using dynamic programming or graph cuts to find optimal seams [36, 65] (see Figure 2(c) and Figure 3(a-c)). In other papers, dense patches are defined such that there is one patch centered at every pixel. In this case, many patches simultaneously overlap, and thus the blending operation is typically done as a weighted average of many different candidate "votes" for colors in the overlap region [7, 96, 106] (see Figure 3(d)).

Synthesis can further be divided into *greedy* algorithms that synthesize pixels or patches only once, and *iterative optimization* algorithms that use multiple passes to repeatedly improve the texture in the output image. The image quilting [36] method shown in Figure 2 is a greedy algorithm because it simply places patches in raster order in a single pass. A limitation of greedy algorithms is that if a mistake is

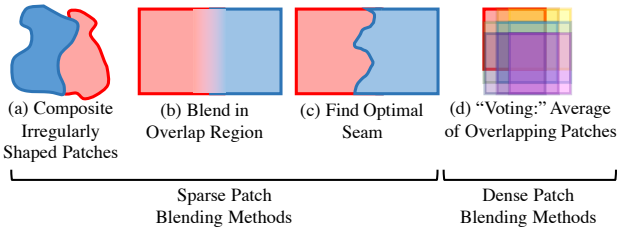


Fig. 3 Blending methods for patch-based synthesis. On the left are shown three methods for blending sparse patches, which are defined as patches having a relatively small overlap region: (a) irregularly-shaped patches can be composited [90]; (b) overlapping patches can be blended in the overlap region [70]; or (c) optimal seams can be found to make a hard "cut" between patches. On the right (d) is shown a method for blending densely overlapping patches, where a patch is defined around every pixel: an average of all overlapping colors is computed by a voting process [96, 106].

made in synthesis, the algorithm cannot later recover from the mistake. In contrast, dense patch synthesis algorithms [7, 59, 96, 106] typically repeat the matching and blending stages as an optimization. In the matching stage, patches within the current estimate for the output image are matched against the exemplar to establish potential improvements to the texture. Next, in the blending stage, these texture patches are copied and blended together by the "voting" process to give an improved estimate for the output image at the next iteration. Iterative optimization methods also typically work in a *coarse-to-fine manner*, by running the optimization on an image pyramid [18]: the optimization is repeated until convergence at a coarse resolution, and then this is repeated at successively finer resolutions until the target image resolution is reached.

4 Matching algorithms

In this section, we discuss algorithms for finding good matches between the output image being synthesized and the input exemplar image. As explained in the previous section, patch-based synthesis algorithms have two main stages: *matching* and *blending*. The matching stage locates the best patches to copy from the exemplar image to the output image that is being synthesized.

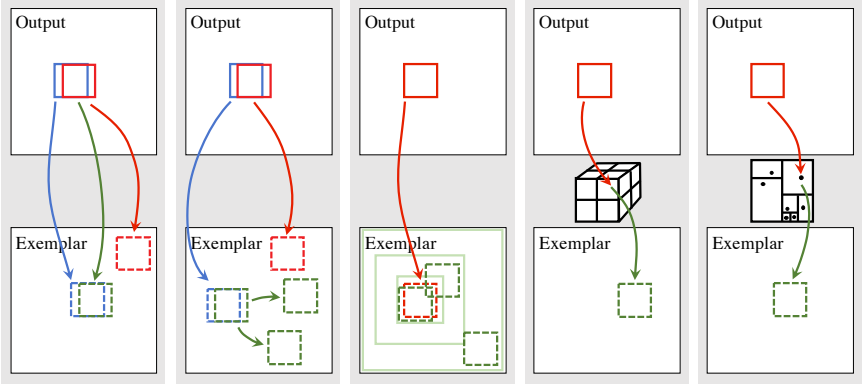
Matching is generally done by minimizing a distance term between a partially or completely synthesized patch in the output image and a same-shaped region in the exemplar image: a search is used to find the exemplar patch that minimizes this distance. We call the distance a *patch distance* or *neighborhood distance*. For example, in the image quilting [36] example of Figure 2, this distance is defined by measuring the L^2 norm between corresponding pixel colors in the overlap region between blocks B1 and B2. For iterative optimization methods [96, 106], the patch distance is frequently defined as an L^2 norm between corresponding pixel colors of a *pxp* square patch in the output image and the same-sized region in the exemplar image. More generally, the patch distance could potentially operate on any image features (e.g. SIFT features computed densely on each pixel [73]), use any function to measure the error of the match (including functions not satisfying the triangle inequality), and use two or more degrees of freedom for the search over correspondences from the output image patch and the exemplar patch (for example, in addition to searching to find the (x, y) translational coordinates of an exemplar patch to match, a rotation angle θ, and scale s could also be searched over).

Typically, matching then proceeds by finding nearest neighbors from the synthesized output S to the exemplar E according to the patch distance. In the terminology of Barnes et al. [8], for the case of two (XY) translational degrees of freedom, we can define a *Nearest Neighbor Field (NNF)* as a function $f : S \mapsto \mathbb{R}^2$ of offsets, defined over all possible patch coordinates (locations of patch centers) in image S, mapping to the center coordinate of the corresponding most similar patch in the exemplar.

Although search in the matching stage can be done in a brute-force manner by exhaustively sampling the parameter space, this tends to be so inefficient as to be impractical for all but the smallest exemplar images. Thus, much research has been devoted to more efficient matching algorithms. Generally, research has focused on approximation algorithms for the matching, because exact algorithms remain slower [107], and the human visual system is not sensitive to small errors in color.

We will first discuss different approximate matching algorithms for patches, followed by a discussion of how these can be generalized to apply to correspondence finding algorithms in computer vision. In Figure 4 are shown illustrations of the key components of a number of approximate matching algorithms. We now discuss five matching algorithms.

Matching using coherence. One simple technique to find good correspondences is to take advantage of *coherence* [3]. Typically, the nearest neighbor field (NNF) used for matching is initialized in some manner, such as by random sampling. However, random correspondences are quite poor, that is, they have high approximation error. An illustration of



(a) Coherence (Propagation) [3] (b) k-coherence [101] (c) Random search [7] (d) Locality sensitive hashing [9, 63] (e) kd-trees [46, 87]

Fig. 4 The key components of different approximate matching algorithms. The goal of each matching algorithm is to establish correspondences between patches in the output image and patches in the exemplar. Each algorithm proposes one or more candidate matches, out of which is selected the match with minimal patch distance. See the body of Section 4 for a discussion of each algorithm.

coherence is shown in Figure 4(a). Suppose that during synthesis, we are examining a current patch in the output image (shown as a solid red square) and it has a poor correspondence in the exemplar (shown as a dotted red square, with the red arrow indicating the correspondence), that is, the correspondence has high patch distance. In this case, a better correspondence might be obtained from an adjacent patch such as the blue one to the left. However, the correspondence from the adjacent left patch (in blue) must be shifted to the right to obtain an appropriate correspondence for the current patch. The new (shifted) candidate correspondence is shown in green. The patch distance is evaluated for this new candidate correspondence, compared with the existing patch distance for the red correspondence, and whichever correspondence has lower patch distance is written back to the NNF. We refer to this process of choosing the best correspondence as *improving a correspondence.* Matching using coherence is also known as *propagation*, because it has the effect of propagating good correspondences across the image. Typically, one might propagate from a larger set of adjacent patches than just the left one: for example, if one is synthesizing in raster order, one might propagate from the left patch, the above patch, and the above-left patch.

Matching using k-coherence. An effective technique for finding correspondences in small, repetitive textures is *k-coherence* [101]. This combines the previous *coherence* technique with precomputed k-nearest neighbors within the exemplar. The k-coherence method is illustrated in Figure 4(b). A precomputation is run on the exemplar, which determines for each patch in the exemplar (illustrated by a green square), what are the k most similar patches located elsewhere in the exemplar (illustrated in green, for $k = 2$). Now suppose that during synthesis, similarly as before, we are examining a current patch, which has a poor correspondence, shown in red. We first apply the coherence rule: we look up the adjacent left patch's correspondence within the exemplar (shown in blue), and shift the exemplar patch to the right by one pixel to obtain the coherence candidate (shown in green). The coherence candidate as well as its k nearest neighbor candidates (all shown as green squares) are all considered as candidates to improve the current patch's correspondence. This process can be repeated for other coherence candidates, such as for the above patch.

Matching using PatchMatch. When real-world photographs are used, k-coherence alone is insufficient to find good patch correspondences, because it assumes that the image is a relatively small and repetitive texture. PatchMatch [6–8] allows for better global correspondences within real-world photographs. It augments the previous coherence (or propagation) stage with a random search process, which can search for good correspondences across the entire exemplar image, but places most samples in local neighborhoods surrounding the current correspondence. Specifically, the candidate correspondences are sampled uniformly within sampling windows that are centered on the best current correspondence. As each patch is visited, the sampling window initially has the same size as the exemplar image, but it then contracts in width

and height by powers of two until the it reaches 1 pixel in size. This random search process is shown in Figure 4(c). Unlike methods such as locality-sensitive hashing (LSH) or kd-trees, PatchMatch takes less memory, and it is more flexible: it can use an arbitrary patch distance function. The Generalized PatchMatch algorithm [8] can utilize arbitrary degrees of freedom, such as matching over rotations and scales, and can find approximate k-nearest neighbors instead of only a single nearest neighbor. For the specific case of matching patches using 2D translational degrees of freedom only under the L^2 norm, PatchMatch is more efficient than kd-trees when they are used naively [7], however, it is less efficient than state-of-the-art techniques that combine LSH or kd-trees with coherence [46, 63, 87]. Recently, graph-based matching algorithms have also been developed based on PatchMatch, which operate across image collections: we discuss these next in Section 6.

Matching using locality sensitive hashing. Locality-sensitive hashing [29] (LSH) is a dimension reduction and quantization method that maps from a high dimensional feature space down to a lower-dimensional quantized space that is suitable to use as a "bucket" in a multidimensional hash table. For example, in the case of patch synthesis, the feature space for a *pxp* square patch in RGB color space could be $\mathbb{R}^{3p^2}$, because we could stack the RGB colors from each of the p^2 pixels in the patch into a large vector. The hash bucket space could be some lower-dimensional space such as $\mathbb{N}^6$. The "locality" property of LSH is that similar features map to the same hash table bucket with high probability. This allows one to store and retrieve similar patches by a simple hash-function lookup. One example of a locality-sensitive hashing function is a projection onto a random hyperplane followed by quantization [29]. Two recent works on matching patches using LSH are coherency-sensitive hashing (CSH) [63] and PatchTable [9]. These two works have a similar hashing process and both use coherence to accelerate the search. Here we discuss the hashing process of PatchTable, because it computes matches only from the output image to the exemplar, whereas CSH computes matches from one image to the other and vice versa.

The patch search process for PatchTable [9] is illustrated in Figure 4(d). First, in a precomputation stage, a multidimensional hash table is created, which maps a hash bucket to an exemplar patch location. The exemplar patches are then inserted into the hash table. Second, as shown in Figure 4(d), during patch synthesis, for each output patch, we use LSH to map the patch to a hash table cell, which stores the location of an exemplar patch. Thus, for each output patch, we can look up a similar exemplar patch. In practice, this hash lookup operation is done only on a sparse grid for efficiency, and coherence is used to fill in the gaps.

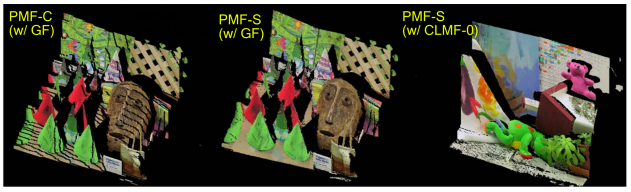


Fig. 5 Results from PatchMatch filter [77]. The algorithm accepts a stereo image pair as input, and estimates stereo disparity maps. The resulting images show a rendering from a new 3D viewpoint. Reproduced with permission from [77], © 2013.

Matching using tree-based techniques. Tree search techniques have long been used in patch synthesis [46, 70, 87, 96, 105]. The basic idea is to first reduce the dimensionality of patches using a technique such as principal components analysis (PCA) [35], and then insert the reduced dimensionality feature vectors into a tree data structure that adaptively divides up the patch appearance space. The matching process is illustrated in Figure 4(e). Here a kd-tree is shown as a data structure that indexes the patches. A kd-tree is an adaptive space-partitioning data structure that divides up space by axis-aligned hyperplanes to conform to the density of the points that were inserted into the tree [85]. Such kd-tree methods are the state-of-the-art for efficient patch searches for 2D translational matching with L^2 norm patch distance function [46, 87].

We now review the state-of-the-art technique of TreeCANN [87]. TreeCANN works by first inserting all exemplar patches that lie along a sparse grid into a kd-tree. The sparse grid is used to improve the efficiency of the algorithm, because kd-tree operations are not highly efficient, and are memory intensive. Next, during synthesis, output patches that lie along a sparse grid are searched against the kd-tree. Locations between sparse grid pixels are filled using coherence.

Correspondences in computer vision. The PatchMatch [8] algorithm permits arbitrary patch distances with arbitrary degrees of freedom. Many papers in computer vision have therefore adapted PatchMatch to handle challenging correspondence problems such as stereo matching and optical flow. We review a few representative papers here. Bleyer et al. [14] showed that better correspondences

can be found between stereo image pairs by adding additional degrees of freedom to patches so they can tilt in 3D out of the camera plane. The PatchMatch filter work [77] showed that edge-aware filters on cost volumes can be used in combination with PatchMatch to solve labeling problems such as optical flow and stereo matching. Results for stereo matching are shown in Figure 5. Similarly, optical flow [22] with large displacements has been addressed by computing a NNF from PatchMatch, which provides approximate correspondences, and then using robust model fitting and motion segmentation to eliminate outliers. Belief propagation techniques have also been integrated with PatchMatch [12] to regularize the correspondence fields it produces.

5 Images

The patch-based matching algorithms of Section 4 facilitate many applications in image and video manipulation. In this section, we discuss some of these applications. Patch-based methods can be used to inpaint targeted regions or "reshuffle" content in images. They can also be used to edit repeated elements in images. Researchers have also proposed editing and enhancement methods for image collections and videos. These applications incorporate the efficient patch query techniques from Section 4 in order to make running times be practical.

Inpainting and reshuffling. One compelling application of patch-based querying and synthesis methods is image inpainting [43]. Image inpainting removes a foreground region from an image by replacing it with background material found elsewhere in the image, or in other images. In [7], PatchMatch was shown to be useful for interactive high-quality image inpainting, by using an iterative optimization method that works in a coarse-to-fine manner. Specifically, this process works by repeatedly matching partially completed features inside the hole region to better matches in the background region. Various subsequent methods took a similar approach and focused on improving the completion quality for more complex conditions, such as maintaining geometrical information that can preserve continuous structures between the hole to inpaint and existing content [19, 53]. To combine such different strategies into a uniform framework, Bugeau et al. [17] and Arias et al. [2] separately proposed variational systems that can choose different metrics when matching patches to adapt to different kinds of inpainting inputs. Kopf et al. [62] also proposed a method to predict the inpainting quality, which can help to choose the most suitable image inpainting methods for different situations. Most recently, deep learning has been introduced in [88] to estimate the missing patches and produce a feature descriptor. The patch-based approach has also been used for Internet photo applications like image inpainting for street view generation [115]. Image "reshuffling" is the problem where a user roughly grabs a region of an image and moves it to a new position. The goal is that the computer can synthesize a new image consistent with the user constraints. Reshuffling can be treated as the extension of image inpainting techniques, by initializing the regions to be synthesized by user specified contents [26]. In Barnes et al. [7], the reshuffling results are made more controllable by adding user constraints to the PatchMatch algorithm.

Editing repeated elements in images. To perform object level image editing, it is necessary to deform objects, and inpaint occluded parts of the objects and the background. Patch-based methods can be used to address these challenges. One scenario for object-level image manipulation involves repeated elements in textures and natural images. The idea of editing repeated elements was first proposed by the work of RepFinder [23]. In their interactive system, the repeated objects are first detected and extracted. Next, the edited objects are composited on the completed background using a PatchMatch-based inpainting method. One result is shown in Figure 6. Huang et al. [52] improve the selection part of RepFinder and also demonstrate similar editing applications. In addition to exploring traditional image operations like moving, deleting and deforming the repeated elements, novel editing tools were also proposed by Huang et al. In the work of ImageAdmixture [110], mixtures between groups of similar elements were created, as shown in Fig. 6. To create natural appearance in the mixed elements' boundary regions, patch-based synthesis is used to generate the appearance using the pixels from boundaries of other elements within the same group.

Denoising. The Generalized PatchMatch work [8] showed that non-local patch searches can be integrated into the non-local means method [16] for image denoising to improve noise reduction. Specifically, this process works by finding for each image patch, the k most similar matches both globally across the image, and locally within a search region. A weighted average of these patches can be taken to remove noise from the input image patches. Liu and Freeman [71] showed that a similar non-local patch search can be used for video, where optical flow guides the patch search process. The

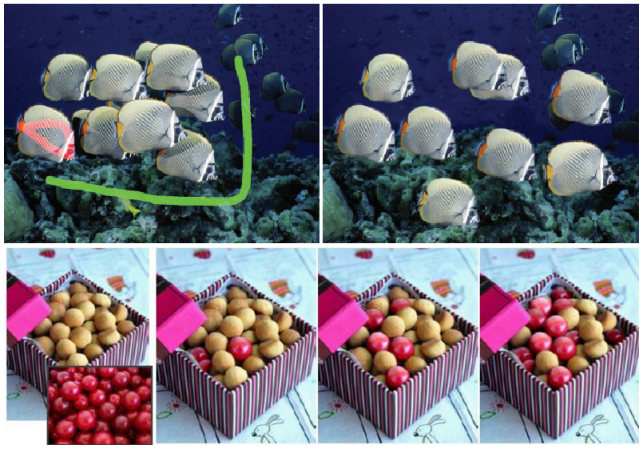


Fig. 6 An illustration of editing repeated elements in an image. Upper row: RepFinder [23] uses patch-based methods to complete the background of the editing results before compositing different layers of foreground objects. Lower row: ImageAdmixture [110] uses patch-based synthesis to determine boundary appearance when generating mixtures of objects. Reproduced with permission from [23], © Associating for Computing Machinery 2010, and [110], © IEEE 2012.

method of Deledalle et al. [30] showed that principal components analysis (PCA) can be combined with patch-based denoising to produce high-quality image denoising results. Finally, Chatterjee and Milanfar [20] showed that a patch-based Wiener filter can be used to reduce noise by exploiting redundancies at the patch level. In the closely related task of image smoothing, researchers have also investigated patch-based methods that use second order feature statistics [58, 72].

6 Image collections

In the scenario where a photographer has multiple images, there are two categories of existing works on utilizing patches as the basic operating units. In the first category, researchers extended patch-based techniques that had previously been applied to single images to multi-view images and image collections as Tong et al. did in [100]. In the second category, patches are treated as a bridges to build connections between different images. We now discuss the first category of editing tools. For stereo images, patch-based inpainting methods have been used to complete regions that have missing pixel colors caused by dis-occlusions when synthesizing novel perspective viewpoints [28, 60, 103]. In the work of Wang et al. [103], depth information is also utilized to aid the patch-based hole filling process for object removal. Morse et al. [84] proposed an extension of PatchMatch to obtain better stereo image inpainting results. Morse et al. complete the depth information first and then add this depth information to

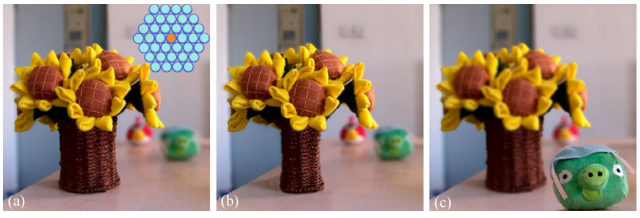


Fig. 7 Results of reshuffling and re-layering the content in a light field by PlenoPatch [111]. Reproduced with permission from [111], © IEEE 2016.

Fig. 8 Matching regions (right) by NRDC [44] of the two input images (left and middle) with different color theme. Reproduced with permission from [44], © Association for Computing Machinery 2011.

PatchMatch's propagation step when finding matches for inpainting both stereo views. Inspired by the commercial development of light field capture devices like PiCam and Lytro, researchers have also developed editing tools for light fields similar to existing 2D image editing tools [112]. One way to look at light fields is as an image array with many different camera viewpoints. Zhang et al. [111] demonstrated a layered patch-based synthesis system which is designed to manipulate light fields as an image array. This enables users to perform inpainting, and re-arrange and re-layer the content in the light field as shown in Fig. 7. In these methods, patch querying speed is a bottleneck in the performance. Thus, Barnes et al. [9] proposed to use a fast query method to accelerate the matching process across all the image collections. The proposed patch-based applications, such as image stitching using a small album and light-field super-resolution were reported to be significantly faster.

In the second category, patches are treated as a bridge to build connections between contents from different images. Unlike previous region matching methods which find similar shapes or global features, these methods focus on matching contiguous regions with similar appearance. This allows a dense, non-rigid correspondence to be estimated between related regions. Non-rigid dense correspondence (NRDC) [44] is a representative work. Based on Generalized

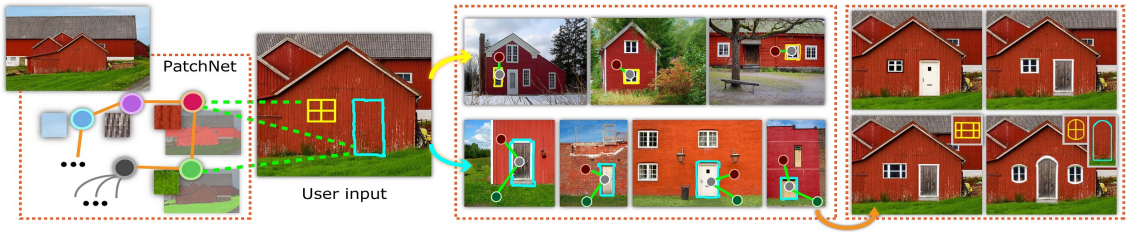


Fig. 9 PatchNet [51] can be used to find a graph that connects local regions, where each local region has an internal repeating texture. PatchNet can be used for library-driven editing applications such as image composition. Reproduced with permission from Hu et al. [51], © Association for Computing Machinery 2013.

PatchMatch [8], the NRDC paper proposed a method to find contiguous matching regions between two related images, by checking and merging the good matches between neighbors. NRDC demonstrated good matching results even when there is a large change in colors or lighting between input images. An example is shown in Fig. 8. The approach of NRDC was further improved and applied to matching contents across an image collection in the work of HaCohen et al. [45]. For large image datasets, Gould et al. [41] proposed a method to build a matching graph using PatchMatch, and optimize a conditional Markov random field to propagate pixel labels to all images from just a small subset of annotated images. Patches are also used as a representative feature for matching local contiguous regions in PatchNet [51]. In this work, an image region with coherent appearance is summarized by a graph node, associated with a single representative patch, while geometric relationships between different regions are encoded by labelled graph edges giving contextual information. As shown in Fig. 9, the representative patches and the contextual information are combined to find reasonable local regions and objects for the purpose of library-driven editing.

7 Video

Here we discuss how patch-based methods can be extended for applications on videos, by incorporating temporal information into patch-based optimizations. We will start by discussing a relatively easier application to high-dynamic range video [57], and then proceed to briefly discuss two works on video inpainting [42, 86], followed by video summarization by means of "video tapestries" [5]. The method of Kalantari et al. [57] reconstructs *high-dynamic range* (HDR) video. A brief definition of HDR imaging is that it extends conventional photography by using computation to achieve greater dynamic range in luminance, typically by using several photographs of the same scene with varying exposure. In Kalantari et al. [57], HDR video is reconstructed by a special video camera that can alternate the exposure at each frame of the video. For example, the exposure of frame 1 could be low, frame 2 could be medium, and frame 3 could be high, and then this pattern could repeat. The goal of the high-dynamic range reconstruction problem then is to reconstruct the missing exposure information: for example, on frame 1 we need to reconstruct the missing medium and high exposure information. This will allow us to reconstruct a video that has high-dynamic range at every frame, and will thus allow us to take high-quality videos that simultaneously include both very dark and bright regions. One solution to this problem is to use optical flow to simply guide the missing information from past and future frames. However, optical flow is not always accurate, so better results are obtained by Kalantari et al. [57] by formulating the problem as a patch-based optimization that fills in missing pixels by minimizing both optical-flow like terms and patch similarity terms. This problem is fairly well-constrained, because there is one constraint image at every frame. Results are shown in Figure 10. Note that Sen et al. [94] also presented a similar patch-based method of HDR reconstruction where the goal is to produce a single output image rather than a video.

The problem of video inpainting, in contrast, is very unconstrained. In video inpainting, the user selects a spacetime region of the video to remove, and then the computer must synthesize an entire volume of pixels in that region that plausibly removes the target objects. The problem is further complicated because both the camera and foreground objects may move, and introduce parallax, occlusions, disocclusions, and shadows. Granados et al. [42] developed a video inpainting method that aligns other candidate frames to the frame that is to be removed, selects among candidate pixels for the

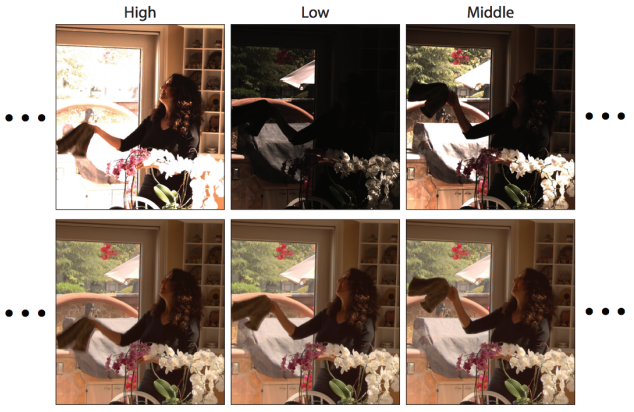


Fig. 10 Results for high-dynamic range (HDR) video reconstruction from Kalantari et al. [57]. On the top row are shown frames recorded by a special video camera that alternates exposure at each frame (between high, low, and middle exposure). At bottom is shown the HDR reconstruction, which has both light and dark areas well-exposed. See discussion in Section 7. Reproduced with permission from [57], © Association for Computing Machinery 2013.

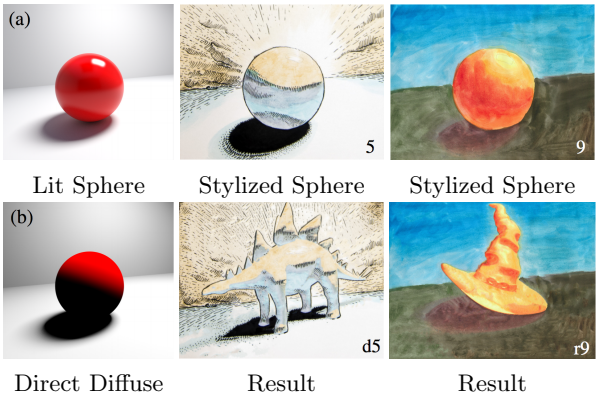


Fig. 11 Results for StyLit [39]. A sphere is rendered using global illumination (upper left), and decomposed into different lighting channels such as direct diffuse channel (lower left) by light path expressions [48]. An artist stylizes the sphere in the desired target style (upper row, middle and right), and this can be transferred on to a 3D rendered model automatically using patch-based synthesis (bottom row, middle and right). See Section 8 for discussion. Reproduced with permission from [39] © Association for Computing Machinery 2016.

inpainting using a color-consistency term, and then removes intensity differences using gradient-domain fusion. Granados et al. assume a piecewise planar background region. Newson et al. [86] inpaint video by using a global, patch-based optimization. Specifically, Newson et al. introduce a spatio-temporal extension to PatchMatch to accelerate the search problem, use a multi-resolution texture feature pyramid to improve texture, and estimate background movement using an affine model.

Finally, the video tapestries [5] work shows that patch-based methods can also be applied to producing pleasing summaries of video. Video tapestries are produced by selecting a hierarchical set of keyframes, with one keyframe level per zoom level, so that a user can interactively zoom in to the tapestries. An appealing summary is then produced by compacting or summarizing the resulting layout image so that it is slightly smaller along each dimension: this facilitates the joining of similar regions, and the removal of repetitive features.

Deblurring. Here, we discuss the application of patch-based methods to one hard inverse problem in computer vision, deblurring. Recently, patch-based techniques have been used to deblur images that are blurred due to say camera shake. Cho et al. [24] showed video can be deblurred by observing that some frames are sharper than others. The sharp regions can be detected and used to restore blurry regions in nearby frames. This is done by a patch-based synthesis process that ensures spatial and temporal coherence. Sun et al. [98] showed that blur kernels can be estimated by using a patch prior that is customized towards modeling corner and edge regions. The blur kernel and the deblurred image can both be estimated by an iterative process by imposing this patch prior. Sun et al. [99] later investigated whether the deblurring results could be improved by training on similar images. Sun et al. [99] showed that deblurring results could be improved if patch priors that locally adapt based on region correspondences are used, or multi-scale patch-pyramid priors are used.

8 Synthesis for specialized domains

In this section, we investigate synthesis algorithms that are tailored towards specialized problem domains, including mimicking artistic style, synthesis of artistic brushes, decorative patterns, synthesis for 3D and 3D fabrication, fluid synthesis, and synthesis of rich materials.

Mimicking artistic style. Patch-based synthesis can mimic the style of artists, such as oil paint style, watercolor style, or even abstract styles. We discuss an early work in this area, image analogies [49], followed by two recent works, Bénard et al. [11] and StyLit, which perform example-based stylization using patches. The image analogies framework [49] gave early results showing the transfer of oil paint and watercolor styles. This style transfer works by providing an exemplar photograph A, a stylized variant of the exemplar A', and an input photograph B. The image analogies

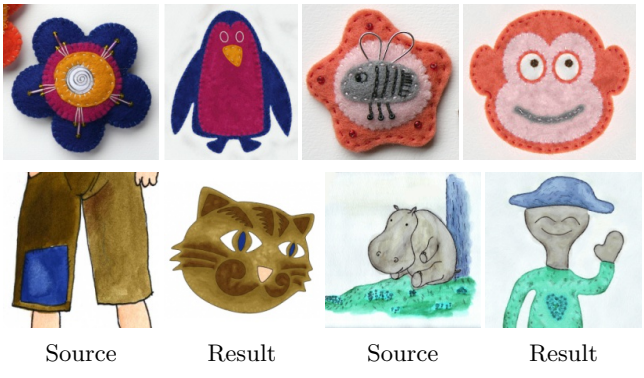


Fig. 12 Results for painting by feature [79]. See Section 8 for discussion. Reproduced with permission from [79], © Association for Computing Machinery 2013.

Fig. 13 Results from RealBrush [74]. At left: physically painted strokes are captured with a camera to create a digital library of natural media. At right: an artist has created a digital painting. RealBrush synthesizes realistic brush texture in the digital painting by sampling from the oil paint and plasticine exemplars in the library. Reproduced with permission from [74], © Association for Computing Machinery 2013.

framework then predicts a stylized version B' of the input photograph B. Image parsing was shown by [109] to improve results for such exemplar-based stylization. The method of Bénard et al. [11] allows artists to paint over certain keyframes in a 3D rendered animation. Other frames are automatically stylized in the target style by using a temporally coherent extension of image analogies to the video cube. The StyLit method [39] uses the "lit sphere" paradigm [97] for transfer of artistic style. In this approach, a well-lit photorealistic sphere is presented to the artist, so that the artist can produce a non-photorealistic examplar of the same sphere. In StyLit [39], the sphere is rendered using global light transport, and the rendered image is decomposed into lighting channels such as direct diffuse illumination, direct specular illumination, first two bounces of diffuse illumination, and so forth. This is augmented by an iterative patch assignment process, which avoids producing bad matches while also avoiding excessive regularity in the produced texture. See Figure 11 for example stylized results.

Synthesis of artistic brushes and decorative patterns. Patch-based synthesis methods can easily generate complex textures and structures. For this reason, they have been adopted for artistic design. We first discuss digital brush tools, and then discuss tools for designing decorative patterns.

Research on digital brushes includes methods for painting directly from casually-captured texture exemplars [78, 79, 92], RealBrush [74], which synthesizes from carefully captured natural media exemplars, and "autocomplete" of repetitive strokes in paintings [108]. Ritter et al. [92] initially developed a framework that allows artists to paint with textures on a target image by sampling textures directly from an exemplar image. This approach works by adapting typical texture energy functions to specially handle boundaries and layering effects. Painting by feature [79] further advanced the area of texture-based painting. They improved the sampling of patches along boundary curves, where humans are particularly sensitive to misalignments or repetitions of texture, and then filled other areas using patch-based inpainting. Results are shown in Figure 12. Brushables [78] also improved upon the example-based painting approach by allowing the artist to specify a direction field for the brush, and simultaneously synthesized both the edge and interior regions. RealBrush [74] allows for rich natural media to be captured with a camera, processed with fairly minimal user input, and then used in subsequent digital paintings. Results from RealBrush are shown in Figure 13. The autocompletion of repetitive painting strokes [108] works by detecting repetitive painting operations, and suggesting an autocompletion to users if repetition is detected. Strokes are represented in a curve representation which is sampled so that neighborhoods matching can be performed between collections of samples.

Decorative patterns are frequently used in illustrated manuscripts, formal invitations, web pages, and interior design. We discuss several recent works on synthesis of decorative patterns. DecoBrush [75] allows a designer to synthesize decorative patterns by giving the algorithm examples of the patterns, and specifying a path that the pattern should follow. See Figure 14 for DecoBrush results, including the pattern exemplars, a path, and a resulting decorative pattern. In cases where the pattern is fairly regular and self-similar, Zhou et al. [114] showed that a simpler dynamic programming technique can be used to synthesize patterns along curves. Later, Zhou et al. [113] showed that decorative patterns can be synthesized

Fig. 14 Results from DecoBrush [75]. See discussion in Section 8. Reproduced with permission from [75], ©Association for Computing Machinery 2014.

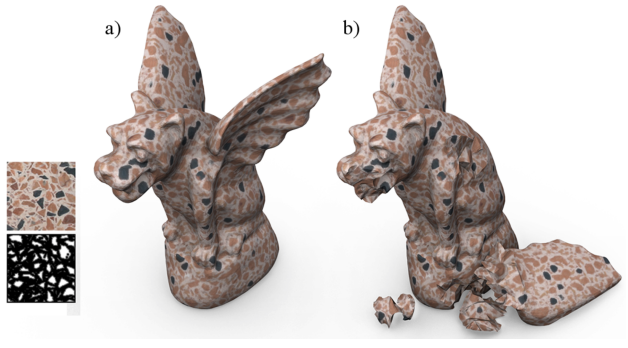


Fig. 15 Lazy solid texture synthesis [33] can be used to synthesize 3D texture from a 2D exemplar. This is done in a *lazy* manner, so that only voxels near the surface of a 3D model need be synthesized. At left is the exemplar. (a) a result for a 3D mesh, (b) a consistent texture can be generated even if the mesh is fractured. Reproduced with permission from [33], © The Author(s) Journal compilation 2008, © The Eurographics Association and Blackwell Publishing 2008.

with better controls over topology (e.g. number of holes and connected components), by first synthesizing the topology using a topological descriptor, and then synthesizing the pattern itself. This latter work also demonstrated design of 3D patterns, which we discuss next.

3D synthesis. We first discuss synthesis of 3D structures that do not need to be physically fabricated. Early texture synthesis works focused on synthesizing voxels for the purposes of synthesizing geometry on surfaces [13] or synthesizing 3D volumetric textures [33, 61]. For volumetric 3D texture synthesis, Kopf et al. [61] showed that 3D texture could be effectively synthesized from a 2D exemplar by matching 2D neighborhoods aligned along the three principal axes in 3D space. Kopf et al. [61] additionally showed that histogram matching can make the synthesized statistics more similar to the exemplar. Dong et al. [33] showed that such 3D textures can be synthesized in a lazy manner, such that if a surface is to be textured, then only voxels near the surface need be evaluated. A key observation in their paper is to synthesize the 3D volume from precomputed sets of 3D candidates, each of which is a triple of interleaved 2D neighborhoods. This reduces the search space during synthesis. See Figure 15 for some results. In a similar manner, Lee et al. [66] showed that arbitrary 3D geometry could be synthesized. In their case, to reduce the computational burden, rather than synthesizing voxels directly, they synthesized an adaptive signed distance field using an octree representation.

A separate line of research focused on the synthesis of discrete vector elements in 3D [81, 82]. In Ma et al. 2011 [82], collections of discrete 3D elements are synthesized by matching neighborhoods to a reference exemplar of 3D elements. Multiple samples can be placed along each 3D element, which allows for synthesis of oblong objects such as a bowl of spaghetti or bean vegetables. Constraints can be incorporated into the synthesis process, such as physics, boundary constraints, or orientation fields. Later, Ma et al. 2013 [81] extended this synthesis method to handle dynamically animating 3D objects, such as groups of fish or animating noodles.

Synthesis for 3D fabrication. Physical artifacts can be fabricated in a computational manner by using subtractive manufacturing, such as computer-numerical control (CNC) milling and laser cutters, or using additive manufacturing, such as fused deposition modeling (FDM) printing and photopolymerization. During the printing process, it may be desirable to add fine-scale texture detail, such as ornamentation by flowers or swirls on a lampshade. Researchers have explored using patch-based and by-example methods in this direction recently. Dumas et al. [34] showed that a mechanical optimization technique called *topology optimization* can be combined with an appearance optimization that controls patterns and textures. *Topology optimization* designs 2D or 3D shapes so as to have structural properties such as supporting force loads. Subsequently, Martínez et al. [83] showed that geometric patterns including empty regions can be synthesized along the surfaces of 3D shapes such that the printed shape is structurally sound. The method works by a joint optimization of a patch-based appearance term and a structural soundness term. The synthesized patterns follow the input exemplar, as shown in Figure 16 (left). Recently, Chen et al. [21] demonstrated that fine filigrees can be fabricated in an example-driven manner. Their method works by

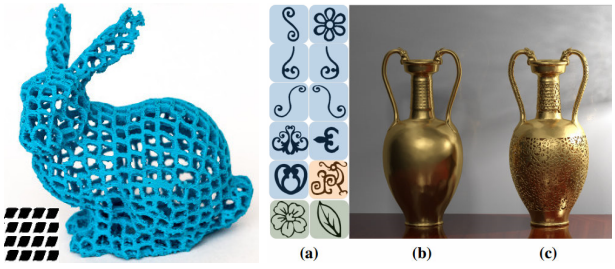


Fig. 16 Synthesis for 3D fabrication. At left are results from Martínez et al. [83]. At right are shown filigrees from Chen et al. [21]. Subtle patterns of material and empty space can be created along the surface of the object, such that they resemble 2D exemplars. See the paper body for more discussion. Reproduced with permission from [21, 83], © Association for Computing Machinery, 2015 and 2016, respectively.

reducing the filigree to a skeleton that references the base elements, and they also relax the problem by permitting partial overlap of elements. Example filigrees are shown in Figure 16 (right).

Fluids. Texture synthesis has been used to texture fluid animations in 2D and 3D. The basic idea is to texture one frame of an animation, then advect the texture forward in time by following the fluid motion, and then re-texture the next frame if needed by starting from this initial guess. Some early works explored these ideas for synthesizing 2D textures directly on 3D fluids [4, 64]. More recently, patch-based fluid synthesis research has focused on stylizing and synthesizing fluid flows that match a target exemplar [15, 55, 80]. Ma et al. [80] focused on synthesizing high-resolution motion fields that match an exemplar, while the low-resolution flow matches a simulation or other guidance field. Browning et al. [15] synthesized stylized 2D fluid animations that match an artist's stylized exemplar. Jamriška et al. [55] implements a more sophisticated system of transfer from exemplars, including support for video exemplars, encouraging uniform usage of exemplar patches, and improvement of the texture when it is advected for a long time. Figure 17 shows a result for fluid texturing with the method of Jamriška et al. [55].

Rich materials. Real-world materials tend to have complex appearance, including varying amounts of weathering, gradual or abrupt transitions between different materials, varying normals, lighting, orientation, and scale. Texture approaches can be used to factor out or manipulate these rich material properties. We discuss four recent works: the first two focus on controlling material properties [1, 27], the third focuses on controlling weathering [10], and the last focuses on capturing spatially varying

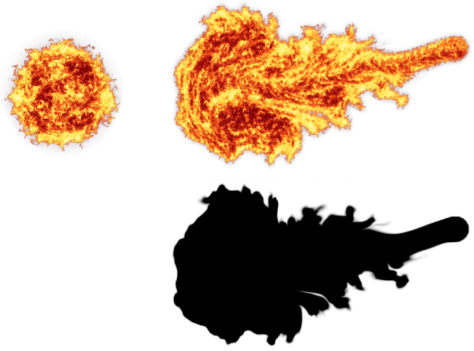


Fig. 17 Results from Jamriška et al. [55]. At left is an exemplar of a fluid (in this case, fire is modeled as a fluid). At bottom is the result of a fluid simulation, used to synthesize a frame of the resulting fluid animation shown at top. See discussion in Section 8. Reproduced with permission from [55], © Association for Computing Machinery 2015.

BRDFs by taking two photographs of a textured material [1]. Image melding [27] permits an artist to create smooth transitions between two source images in a way that gradually transitions between inconsistent colors, textures, and structural properties. This permits applications in object cloning, stitching of challenging panoramas, and hole filling using multiple photographs. Diamanti et al. [32] focuses on a similar problem of synthesizing images or materials that follow user annotations, such as the desired scale, material, orientation, lighting, and so forth. In cases that the desired annotation is not present, Diamanti et al. [32] performs texture interpolation to synthesize a plausible guess for unobserved material properties. A result from Diamanti et al. [32] is shown in Figure 18. The recent work by Bellini et al. [10] demonstrates that automatic control over degree of weathering can be achieved for repetitive, textured images. This method finds for each patch of the input image a measure for how weathered it is, by measuring its dissimilarity to other similar patches. Subsequently, weathering can be reduced or increased in a smooth, time-varying manner by replacing low-weathered patches with high-weathered ones, or vice versa. The method of Aittala et al. [1] uses a flash, no-flash pair of photographs of a textured material to recover a spatially varying BRDF representation for the material. This is done by leveraging a self-similarity observation that although a texture's appearance may vary spatially, for any point on the texture, there exist many other points with similar reflectance properties. Aittala et al. fit a spatially varying BRDF that models diffuse and specular, anisotropic reflectance over a

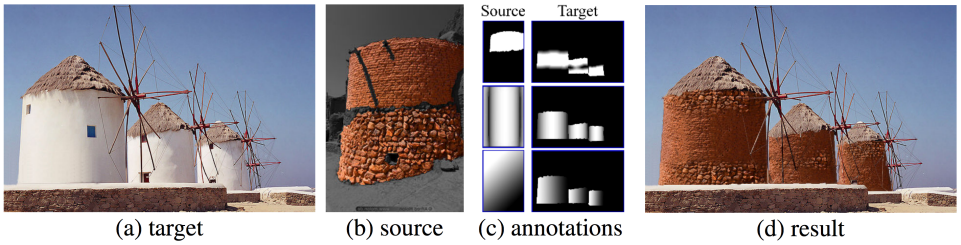


Fig. 18 Results for complex material synthesis from Diamanti et al. [32]. The target image (a) is used as a backdrop and the source (b) is used as an exemplar for the patch-based synthesis process. The artist creates annotations for different properties of the material, such as large vs small bricks, the normal, and lit vs shadow. A result image (d) is synthesized. Here about 50% of desired patch annotations were not seen in the exemplar, and were instead interpolated between the known materials. See discussion in Section 8. Reproduced with permission from [32], © Association for Computing Machinery 2015.

detailed normal map.

9 Discussion

There have been great advances in patch-based synthesis in the last decade. Techniques that originally synthesized larger textures from small ones have been adapted to many domains, including editing of images and image collections, video, denoising and deblurring, synthesis of artistic brushes, decorative patterns, synthesis for 3D fabrication, and fluid stylization. We now discuss one alternative approach that could potentially inspire future work: deep learning for synthesis of texture.

Deep neural networks have recently been used for synthesis of texture [31, 40, 102]. Gatys et al. [40] showed that artistic style transfer can be performed from an exemplar to an arbitrary photograph by means of deep neural networks that are trained on recognition problems for millions of images. Specifically, this approach works by optimizing the image so that feature maps at the higher levels of a neural network have similar pairwise correlations as the exemplar, but the image is still not too dissimilar in its feature map to the input image. This optimization can be carried out by back-propagation. Subsequently, Ulyanov et al. [102] showed that such a texture generation process can be accelerated by pre-training a convolutional network to mimic a given exemplar texture. Denton et al. [31] showed that convolutional neural networks can be used to generate novel images, by training a different convolutional network at each level of a Laplacian pyramid, using a generative adversarial network loss.

Although neural networks are parametric models that are quite different from the non-parametric approach commonly used in patch-based synthesis, it may be interesting in future research to combine the benefits of both approaches. For example, unlike neural networks, non-parametric patch methods tend to use training-free k-nearest neighbor methods, and so they do not need to go through a training process to "learn" a given exemplar. However, neural networks have recently shown state-of-the-art performance on many hard inverse problems in computer vision such as semantic segmentation. Thus, hybrid approaches might be developed that take advantage of the benefits of both techniques.

Acknowledgements

Thanks to Professor Shi-Min Hu at Tsinghua University for his support in this project. Thanks to the National Science Foundation for support under grants CCF 0811493 and CCF 0747220. Thanks for support from the General Financial Grant from the China Postdoctoral Science Foundation (No. 2015M580100).

References

[1] M. Aittala, T. Weyrich, and J. Lehtinen. Two-shot svbrdf capture for stationary materials. *ACM Transactions on Graphics*, 34(4):110, 2015.

[2] P. Arias, G. Facciolo, V. Caselles, and G. Sapiro. A variational framework for exemplar-based image inpainting. *International journal of computer vision*, 93(3):319–347, 2011.

[3] M. Ashikhmin. Synthesizing natural textures. pages 217–226. ACM, 2001.

[4] A. W. Bargteil, F. Sin, J. E. Michaels, T. G. Goktekin, and J. F. O'Brien. A texture synthesis method for liquid animations. In *Proceedings of the 2006 ACM*

SIGGRAPH/Eurographics symposium on Computer animation, pages 345–351. Eurographics Association, 2006.

[5] C. Barnes, D. B. Goldman, E. Shechtman, and A. Finkelstein. Video tapestries with continuous temporal zoom. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 29(3), Aug. 2010.

[6] C. Barnes, D. B. Goldman, E. Shechtman, and A. Finkelstein. The patchmatch randomized matching algorithm for image manipulation. *Communications of the ACM*, 54(11):103–110, 2011.

[7] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), Aug. 2009.

[8] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein. The generalized PatchMatch correspondence algorithm. In *European Conference on Computer Vision*, Sept. 2010.

[9] C. Barnes, F.-L. Zhang, L. Lou, X. Wu, and S.-M. Hu. Patchtable: Efficient patch queries for large datasets and applications. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, Aug. 2015.

[10] R. Bellini, Y. Kleiman, and D. Cohen-Or. Time-varying weathering in texture space. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, pages 1–18, 2016.

[11] P. Bénard, F. Cole, M. Kass, I. Mordatch, J. Hegarty, M. S. Senn, K. Fleischer, D. Pesare, and K. Breeden. Stylizing Animation By Example. *ACM Transactions on Graphics*, 32(4):119:1–119:12, July 2013.

[12] F. Besse, C. Rother, A. Fitzgibbon, and J. Kautz. Pmbp: Patchmatch belief propagation for correspondence field estimation. *International Journal of Computer Vision*, 110(1):2–13, 2014.

[13] P. Bhat, S. Ingram, and G. Turk. Geometric texture synthesis by example. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 41–44. ACM, 2004.

[14] M. Bleyer, C. Rhemann, and C. Rother. Patchmatch stereo-stereo matching with slanted support windows. In *BMVC*, volume 11, pages 1–11, 2011.

[15] M. Browning, C. Barnes, S. Ritter, and A. Finkelstein. Stylized keyframe animation of fluid simulations. *NPAR 2014, Proceedings of the 12th International Symposium on Non-photorealistic Animation and Rendering*, June 2014.

[16] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65. IEEE, 2005.

[17] A. Bugeau, M. Bertalmío, V. Caselles, and G. Sapiro. A comprehensive framework for image inpainting. *Image Processing, IEEE Transactions on*, 19(10):2634–2645, 2010.

[18] P. J. Burt. Fast filter transform for image processing. *Computer graphics and image processing*, 16(1):20–51, 1981.

[19] F. Cao, Y. Gousseau, S. Masnou, and P. Pérez. Geometrically guided exemplar-based inpainting. *SIAM Journal on Imaging Sciences*, 4(4):1143–1179, 2011.

[20] P. Chatterjee and P. Milanfar. Patch-based near-optimal image denoising. *Image Processing, IEEE Transactions on*, 21(4):1635–1649, 2012.

[21] W. Chen, X. Zhang, S. Xin, Y. Xia, S. Lefebvre, and W. Wang. Synthesis of filigrees for digital fabrication. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, pages 1–18, 2016.

[22] Z. Chen, H. Jin, Z. Lin, S. Cohen, and Y. Wu. Large displacement optical flow from nearest neighbor fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2443–2450, 2013.

[23] M.-M. Cheng, F.-L. Zhang, N. J. Mitra, X. Huang, and S.-M. Hu. Repfinder: finding approximately repeated scene elements for image editing. In *ACM Transactions on Graphics (TOG)*, volume 29, page 83. ACM, 2010.

[24] S. Cho, J. Wang, and S. Lee. Video deblurring for hand-held cameras using patch-based synthesis. *ACM Transactions on Graphics (TOG)*, 31(4):64, 2012.

[25] T. S. Cho, S. Avidan, and W. T. Freeman. A probabilistic image jigsaw puzzle solver. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 183–190. IEEE, 2010.

[26] T. S. Cho, M. Butman, S. Avidan, and W. T. Freeman. The patch transform and its applications to image editing. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[27] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen. Image melding: combining inconsistent images using patch-based synthesis. *ACM Transactions on Graphics (TOG)*, 31(4):82, 2012.

[28] I. Daribo and B. Pesquet-Popescu. Depth-aided image inpainting for novel view synthesis. In *Multimedia Signal Processing (MMSP), 2010 IEEE International Workshop on*, pages 167–170. IEEE, 2010.

[29] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM, 2004.

[30] C.-A. Deledalle, J. Salmon, A. S. Dalalyan, et al. Image denoising with patch based pca: local versus global. In *BMVC*, pages 1–10, 2011.

[31] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems*, pages 1486–1494, 2015.

[32] O. Diamanti, C. Barnes, S. Paris, E. Shechtman,

and O. Sorkine-Hornung. Synthesis of complex image appearance from limited exemplars. In *ACM Transactions on Graphics*, volume 34, Feb. 2015.

[33] Y. Dong, S. Lefebvre, X. Tong, and G. Drettakis. Lazy solid texture synthesis. In *Computer Graphics Forum*, volume 27, pages 1165–1174. Wiley Online Library, 2008.

[34] J. Dumas, A. Lu, S. Lefebvre, J. Wu, T. München, C. Dick, and T. München. By-example synthesis of structurally sound patterns. *ACM Transactions on Graphics (TOG)*, 34(4):137, 2015.

[35] G. H. Dunteman. *Principal components analysis*. Number 69. Sage, 1989.

[36] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346. ACM, 2001.

[37] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038. IEEE, 1999.

[38] I. Fidaner. *A survey on variational image inpainting, texture synthesis and image completion*, 2008. `http://www.scribd.com/doc/3012627/`.

[39] J. Fišer, O. Jamriška, M. Lukáč, E. Shechtman, P. Asente, J. Lu, and D. Sỳkora. Stylit: illumination-guided example-based stylization of 3d renderings. *ACM Transactions on Graphics (TOG)*, 35(4):92, 2016.

[40] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[41] S. Gould and Y. Zhang. Patchmatchgraph: Building a graph of dense patch correspondences for label transfer. In *Computer Vision–ECCV 2012*, pages 439–452. Springer, 2012.

[42] M. Granados, K. I. Kim, J. Tompkin, J. Kautz, and C. Theobalt. Background inpainting for videos with dynamic objects and a free-moving camera. In *Computer Vision–ECCV 2012*, pages 682–695. Springer, 2012.

[43] C. Guillemot and O. Le Meur. Image inpainting: Overview and recent advances. *IEEE signal processing magazine*, 31(1):127–144, 2014.

[44] Y. HaCohen, E. Shechtman, D. B. Goldman, and D. Lischinski. Non-rigid dense correspondence with applications for image enhancement. *ACM transactions on graphics (TOG)*, 30(4):70, 2011.

[45] Y. HaCohen, E. Shechtman, D. B. Goldman, and D. Lischinski. Optimizing color consistency in photo collections. *ACM Transactions on Graphics (TOG)*, 32(4):38, 2013.

[46] K. He and J. Sun. Computing nearest-neighbor fields via propagation-assisted kd-trees. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 111–118. IEEE, 2012.

[47] K. He and J. Sun. Statistics of patch offsets for image completion. In *Computer Vision–ECCV 2012*, pages 16–29. Springer, 2012.

[48] P. S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. *ACM SIGGRAPH Computer Graphics*, 24(4):145–154, 1990.

[49] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340. ACM, 2001.

[50] A. Hosni, C. Rhemann, M. Bleyer, C. Rother, and M. Gelautz. Fast cost-volume filtering for visual correspondence and beyond. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(2):504–511, 2013.

[51] S.-M. Hu, F.-L. Zhang, M. Wang, R. R. Martin, and J. Wang. Patchnet: a patch-based image representation for interactive library-driven image editing. *ACM Transactions on Graphics (TOG)*, 32(6):196, 2013.

[52] H. Huang, L. Zhang, and H.-C. Zhang. Repsnapping: efficient image cutout for repeated scene elements. In *Computer Graphics Forum*, volume 30, pages 2059–2066. Wiley Online Library, 2011.

[53] J.-B. Huang, S. B. Kang, N. Ahuja, and J. Kopf. Image completion using planar structure guidance. *ACM Transactions on Graphics (TOG)*, 33(4):129, 2014.

[54] S. Ilan and A. Shamir. A survey on data-driven video completion. In *Computer Graphics Forum*, volume 34, pages 60–85. Wiley Online Library, 2015.

[55] O. Jamriška, J. Fišer, P. Asente, J. Lu, E. Shechtman, and D. Sỳkora. Lazyfluids: appearance transfer for fluid animations. *ACM Transactions on Graphics (TOG)*, 34(4):92, 2015.

[56] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *arXiv preprint arXiv:1603.08155*, 2016.

[57] N. K. Kalantari, E. Shechtman, C. Barnes, S. Darabi, D. B. Goldman, and P. Sen. Patch-based high dynamic range video. *ACM Transactions on Graphics (Proc. SIGGRAPH ASIA)*, 32(5), Dec. 2013.

[58] L. Karacan, E. Erdem, and A. Erdem. Structure-preserving image smoothing via region covariances. *ACM Transactions on Graphics (TOG)*, 32(6):176, 2013.

[59] A. Kaspar, B. Neubert, D. Lischinski, M. Pauly, and J. Kopf. Self tuning texture optimization. In *Computer Graphics Forum*, volume 34, pages 349–359. Wiley Online Library, 2015.

[60] C. Kim, A. Hornung, S. Heinzle, W. Matusik, and M. Gross. Multi-perspective stereoscopy from light fields. *ACM Transactions on Graphics (TOG)*, 30(6):190, 2011.

[61] J. Kopf, C.-W. Fu, D. Cohen-Or, O. Deussen, D. Lischinski, and T.-T. Wong. Solid texture synthesis from 2d exemplars. In *ACM Transactions on Graphics (TOG)*, volume 26, page 2. ACM, 2007.

[62] J. Kopf, W. Kienzle, S. Drucker, and S. B. Kang.

Quality prediction for image completion. *ACM Transactions on Graphics (TOG)*, 31(6):131, 2012.

[63] S. Korman and S. Avidan. Coherency sensitive hashing. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1607–1614. IEEE, 2011.

[64] V. Kwatra, D. Adalsteinsson, T. Kim, N. Kwatra, M. Carlson, and M. Lin. Texturing fluids. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):939–952, 2007.

[65] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: image and video synthesis using graph cuts. In *ACM Transactions on Graphics (ToG)*, volume 22, pages 277–286. ACM, 2003.

[66] S.-H. Lee, T. Park, J.-H. Kim, and C.-H. Kim. Adaptive synthesis of distance fields. *IEEE transactions on Visualization and Computer Graphics*, 18(7):1135–1145, 2012.

[67] S. Lefebvre, S. Hornus, and A. Lasram. By-example synthesis of architectural textures. In *ACM Transactions on Graphics (TOG)*, volume 29, page 84. ACM, 2010.

[68] M. Levoy and P. Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42. ACM, 1996.

[69] S. Z. Li. *Markov random field modeling in computer vision*. Springer Science & Business Media, 2012.

[70] L. Liang, C. Liu, Y.-Q. Xu, B. Guo, and H.-Y. Shum. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics (ToG)*, 20(3):127–150, 2001.

[71] C. Liu and W. T. Freeman. A high-quality video denoising algorithm based on reliable motion estimation. In *Computer Vision–ECCV 2010*, pages 706–719. Springer, 2010.

[72] Q. Liu, C. Zhang, Q. Guo, and Y. Zhou. A nonlocal gradient concentration method for image smoothing. *Computational Visual Media*, 1(3):197–209, 2015.

[73] D. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[74] J. Lu, C. Barnes, S. DiVerdi, and A. Finkelstein. Realbrush: Painting with examples of physical media. *ACM Transactions on Graphics (TOG)*, 32(4), 2013.

[75] J. Lu, C. Barnes, C. Wan, P. Asente, R. Mech, and A. Finkelstein. Decobrush: Drawing structured decorative patterns by example. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, Aug. 2014.

[76] J. Lu, P. V. Sander, and A. Finkelstein. Interactive painterly stylization of images, videos and 3d animations. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pages 127–134. ACM, 2010.

[77] J. Lu, H. Yang, D. Min, and M. Do. Patch match filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1854–1861, 2013.

[78] M. Lukáč, J. Fišer, P. Asente, J. Lu, E. Shechtman, and D. Sỳkora. Brushables: Example-based edge-aware directional texture painting. In *Computer Graphics Forum*, volume 34, pages 257–267. Wiley Online Library, 2015.

[79] M. Lukáč, J. Fišer, J.-C. Bazin, O. Jamriška, A. Sorkine-Hornung, and D. Sỳkora. Painting by feature: texture boundaries for example-based image creation. *ACM Transactions on Graphics (TOG)*, 32(4):116, 2013.

[80] C. Ma, L.-Y. Wei, B. Guo, and K. Zhou. Motion field texture synthesis. In *ACM Transactions on Graphics (TOG)*, volume 28, page 110. ACM, 2009.

[81] C. Ma, L.-Y. Wei, S. Lefebvre, and X. Tong. Dynamic element textures. *ACM Transactions on Graphics (TOG)*, 32(4):90, 2013.

[82] C. Ma, L.-Y. Wei, and X. Tong. Discrete element textures. In *ACM Transactions on Graphics (TOG)*, volume 30, page 62. ACM, 2011.

[83] J. Martínez, J. Dumas, S. Lefebvre, and L.-Y. Wei. Structure and appearance optimization for controllable shape design. *ACM Transactions on Graphics (TOG)*, 34(6):229, 2015.

[84] B. Morse, J. Howard, S. Cohen, and B. Price. Patchmatch-based content completion of stereo image pairs. In *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, pages 555–562. IEEE, 2012.

[85] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2, 2009.

[86] A. Newson, A. Almansa, M. Fradet, Y. Gousseau, and P. Pérez. Video inpainting of complex scenes. *SIAM Journal on Imaging Sciences*, 7(4):1993–2019, 2014.

[87] I. Olonetsky and S. Avidan. TreeCANN-kd tree coherence approximate nearest neighbor algorithm. In *ECCV*, pages 602–615. Springer, 2012.

[88] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. *arXiv preprint arXiv:1604.07379*, 2016.

[89] D. Pomeranz, M. Shemesh, and O. Ben-Shahar. A fully automated greedy square jigsaw puzzle solver. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 9–16. IEEE, 2011.

[90] E. Praun, A. Finkelstein, and H. Hoppe. Lapped textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 465–470. ACM Press/Addison-Wesley Publishing Co., 2000.

[91] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[92] L. Ritter, W. Li, B. Curless, M. Agrawala, and D. Salesin. Painting with texture. In *Rendering*

Techniques, pages 371–376, 2006.

[93] A. Selim, M. Elgharib, and L. Doyle. Painting style transfer for head portraits using convolutional neural networks. *ACM Transactions on Graphics (TOG)*, 35(4):129, 2016.

[94] P. Sen, N. K. Kalantari, M. Yaesoubi, S. Darabi, D. B. Goldman, and E. Shechtman. Robust patch-based hdr reconstruction of dynamic scenes. *ACM Trans. Graph.*, 31(6):203, 2012.

[95] E. Shechtman, A. Rav-Acha, M. Irani, and S. Seitz. Regenerative morphing. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 615–622. IEEE, 2010.

[96] D. Simakov, Y. Caspi, E. Shechtman, and M. Irani. Summarizing visual data using bidirectional similarity. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[97] P.-P. J. Sloan, W. Martin, A. Gooch, and B. Gooch. The lit sphere: A model for capturing npr shading from art. In *Graphics interface*, volume 2001, pages 143–150. Citeseer, 2001.

[98] L. Sun, S. Cho, J. Wang, and J. Hays. Edge-based blur kernel estimation using patch priors. In *Computational Photography (ICCP), 2013 IEEE International Conference on*, pages 1–8. IEEE, 2013.

[99] L. Sun, S. Cho, J. Wang, and J. Hays. Good image priors for non-blind deconvolution. In *Computer Vision–ECCV 2014*, pages 231–246. Springer, 2014.

[100] R.-f. Tong, Y. Zhang, and K.-L. Cheng. Stereopasting: interactive composition in stereoscopic images. *IEEE transactions on visualization and computer graphics*, 19(8):1375–1385, 2013.

[101] X. Tong, J. Zhang, L. Liu, X. Wang, B. Guo, and H.-Y. Shum. Synthesis of bidirectional texture functions on arbitrary surfaces. 21(3):665–672, July 2002.

[102] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. *arXiv preprint arXiv:1603.03417*, 2016.

[103] L. Wang, H. Jin, R. Yang, and M. Gong. Stereoscopic inpainting: Joint color and depth completion from stereo images. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[104] L.-Y. Wei, S. Lefebvre, V. Kwatra, and G. Turk. State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR*, pages 93–117. Eurographics Association, 2009.

[105] L.-Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 479–488. ACM Press/Addison-Wesley Publishing Co., 2000.

[106] Y. Wexler, E. Shechtman, and M. Irani. Space-time completion of video. *IEEE Transactions on pattern analysis and machine intelligence*, 29(3):463–476, 2007.

[107] C. Xiao, M. Liu, N. Yongwei, and Z. Dong. Fast exact nearest patch matching for patch-based image editing and processing. *IEEE Transactions on Visualization and Computer Graphics*, 17(8):1122–1134, 2011.

[108] J. Xing, H.-T. Chen, and L.-Y. Wei. Autocomplete painting repetitions. *ACM Transactions on Graphics (TOG)*, 33(6):172, 2014.

[109] K. Zeng, M. Zhao, C. Xiong, and S.-C. Zhu. From image parsing to painterly rendering. *ACM Trans. Graph*, 29(1):2, 2009.

[110] F.-L. Zhang, M.-M. Cheng, J. Jia, and S.-M. Hu. Imageadmixture: putting together dissimilar objects from groups. *Visualization and Computer Graphics, IEEE Transactions on*, 18(11):1849–1857, 2012.

[111] F.-L. Zhang, J. Wang, E. Shechtman, Z.-Y. Zhou, J.-X. Shi, and S.-M. Hu. Plenopatch: Patch-based plenoptic image manipulation. 2016.

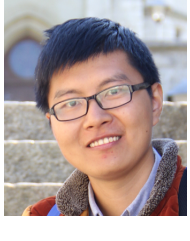
[112] L. Zhang, Y.-H. Zhang, and H. Huang. Efficient variational light field view synthesis for making stereoscopic 3d images. In *Computer Graphics Forum*, volume 34, pages 183–191. Wiley Online Library, 2015.

[113] S. Zhou, C. Jiang, and S. Lefebvre. Topology-constrained synthesis of vector patterns. *ACM Transactions on Graphics (TOG)*, 33(6):215, 2014.

[114] S. Zhou, A. Lasram, and S. Lefebvre. By–example synthesis of curvilinear structured patterns. In *Computer Graphics Forum*, volume 32, pages 355–360. Wiley Online Library, 2013.

[115] Z. Zhu, H.-Z. Huang, Z.-P. Tan, K. Xu, and S.-M. Hu. Faithful completion of images of scenic landmarks using internet images. *IEEE TVCG*, 22(8):1945 – 1958, 2015.

Connelly Barnes is an assistant professor at the University of Virginia. He received his Ph.D. from Princeton University in 2011. His group develops techniques for efficiently manipulating visual data in computer graphics by using semantic information from computer vision. Applications are in computational photography, image editing, art, and hiding visual information. Many computer graphics algorithms are more useful if they are interactive, therefore, his group also has a focus on efficiency and optimization, including some compiler technologies.



Fang-Lue Zhang is a post-doctoral research associate at the department of computer science and technology in Tsinghua University. He received his doctoral degree from Tsinghua University in 2015 and bachelor degree from Zhejiang University in 2009. His research interests include image and video editing, computer vision and computer graphics.