Image Completion Approaches Using the Statistics of Similar Patches

Kaiming He, *Member, IEEE* and Jian Sun, *Member, IEEE*

Abstract—Image completion involves filling missing parts in images. In this paper we address this problem through novel statistics of similar patches. We observe that if we match similar patches in the image and obtain their offsets (relative positions), the statistics of these offsets are sparsely distributed. We further observe that a few dominant offsets provide reliable information for completing the image. Such statistics can be incorporated into both matching-based and graph-based methods for image completion. Experiments show that our method yields better results in various challenging cases, and is faster than existing state-of-the-art methods.

Index Terms—Image completion, image inpainting, natural image statistics

1 Introduction

Image completion involves the issue of filling missing parts in images. This is a non-trivial task in computer vision/graphics: on one hand, the completed images are expected to be visually plausible and has little noticeable artifacts; on the other hand, the algorithm should be efficient, because in practice an image completion tool is often applied with user interactions and needs quick feedbacks. For today's consumer-level multi-mega-pixel cameras, high-quality and fast image completion is still a challenging problem.

One category of image completion methods are diffusion-based [1], [2], [3], [4], [5]. These methods solve Partial Differential Equations (PDE) [1] or similar diffusion systems, so as to propagate colors into the missing regions. They are mainly designed for filling narrow or small holes (also known as "inpainting" [1]). They work less well for large missing regions due to the lack of semantic texture/structure synthesis.

Another category of image completion methods are exemplar-based. They perform more effectively for large holes. In this paper, we further categorize exemplar-based methods into two groups: *matching-based* [6], [7], [8], [9], [10], [11], [12], [13], [14] and *graph-based* [15], [16], [17]. Matching-based methods explicitly match the patches in the unknown region with the patches in the known region, and copy the known content to complete the unknown region. This strategy makes it possible to synthesize textures [6] and more complex structures [7], [8], [9], [10], [11], [12], [13]. Unlike many matching-based methods using greedy fashions, the method proposed by Wexler et al. [11] optimizes a global cost function called the *coherence measure*. This cost function encourages that each patch in the filled region is as similar as possible to a certain known patch. It has been generalized for image retargeting/reshuffling in [14]. This measure helps to yield more coherent results for image completion. But because this cost function inherently has multiple local optima, this method is sensitive to initialization and optimization methods.

Matching patches can be a computationally expensive operation. A fast PatchMatch algorithm [18] largely relieves this problem and is combined with Wexler et al.'s methods [11]. This combination is implemented as the *Content Aware Fill* in Adobe Photoshop [19], is arguably among the current state-of-the-arts in terms of both visual quality and speed.

Besides the matching-based strategy, exemplar-based methods can also be realized by optimizing *graph-based* models like Markov Random Fields (MRFs), as in the works of Priority-BP [15] and Shift-map [17]. Rather than match patches, these methods rearrange the patch/pixel locations to complete the image. The rearrangement is formulated as an MRF, where each node in the graph takes its value among a set of discrete labels. These labels can represent the absolute coordinates of the patches/pixels as in Priority-BP [15], or the relative offsets as in Shift-map [17]. The edges in the graph encourages that the neighboring nodes should have visually coherent content. The MRFs are optimized via well-studied techniques like belief propagation (BP) [20] or graph-cuts [21].

Although avoiding matching patches, the graph-based methods are still computationally expensive: the complexity is linear in the number of labels and also in the number of unknown pixels, so is approximately quadratic in the number of image pixels. Priority-BP [16] adopts label pruning, and Shift-map [17] adopts hierarchical solvers [17]. But they may still take tens of seconds to process small images (e.g., 400×300 pixels).

We note that exemplar-based methods, both *matching-based* and *graph-based*, should implicitly or explicitly assign each unknown pixel/patch an *offset*—the relative location from where it copies the content. Both the coherence measure in Wexler *et al*'s method [11] and the MRFs in Shift-map [17] can be viewed as optimization w.r.t. the offsets (Section 2). But existing methods do not predict reliable

• *K. He and J. Sun are with the Visual Computing Group, Microsoft Research Asia, Beijing 100080, China.*
E-mail: {kahe, jiansun}@microsoft.com.

Manuscript received 23 Apr. 2013; revised 6 Oct. 2013; accepted 1 June 2014. Date of publication 0 . 0000; date of current version 0 . 0000.
Recommended for acceptance by S. Avidan.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2014.2330611

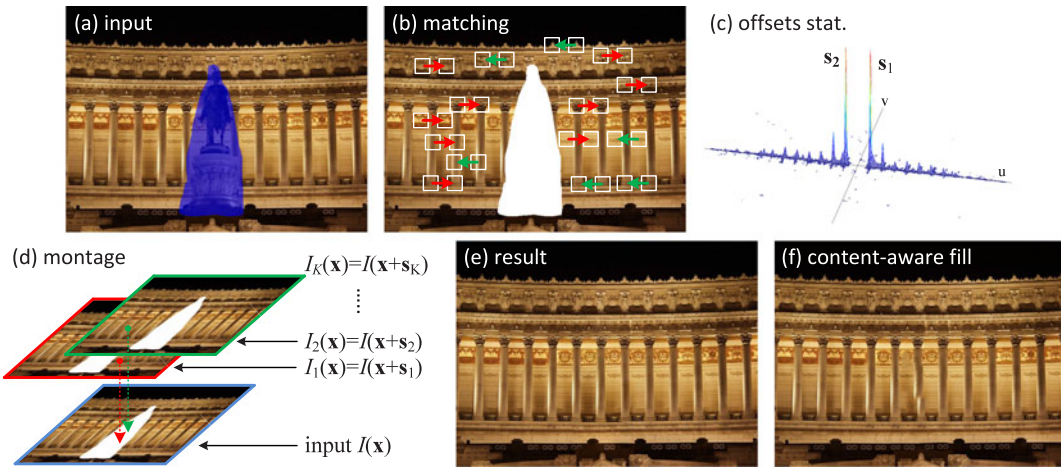


Fig. 1. Outline. (a) Input image with a mask overlayed. (b) Matching similar patches in the known region. (c) The statistics of the offsets of the similar patches. The offsets of the highest peaks are picked out. (d) Combining a set of shifted images with the given offsets. (e) Our graph-based result. (f) Result of Content-Aware Fill.

offsets beforehand, and generally accept all possible offsets in the optimization. We will show that it is beneficial to constrain the offsets using certain statistics of patch offsets. In terms of quality, our constrained offsets may produce better results for both graph-based and matching-based methods. Typically, we find unexpected biases may present for graph-based methods if the offsets are not restricted (*c.f.* Section 3.3). In terms of speed, a small set of pre-defined offsets lead to efficient algorithms.

In this paper, we present novel statistics of similar patches for high-quality and fast image completion. We observe that if we match similar patches in the image, the statistics of patch offsets are sparsely distributed (Fig. 1c): a majority of patches have similar offsets, forming several prominent peaks in the statistics. Such dominant offsets describe how the patterns are most possibly repeated, and thus provide reliable clues for completing the missing region. We observe that these offsets are predictive for completing linear structures, textures, and repeated objects. Then we show that the statistics of patch offsets can be incorporated into both graph-based and matching-based methods for image completion. In our graph-based solution, we create a stack of shifted images corresponding to a few dominant offsets, and combine them via graph-cuts to fill the missing region (Fig. 1d). In our matching-based solution, we optimize the coherence measure but only allow patches to be matched to those shifted by the dominant offsets. In experiments, both methods produce high-quality results in various cases that are challenging for many state-of-the-art methods. The graph-based method is also faster than the competitors including Content-Aware Fill in Adobe Photoshop.

The statistics of patch offsets is a way of representing image self-similarities. In the literature, various definitions of image self-similarities have been investigated for texture synthesis [22], object detection/classification [23], [24], image segmentation [25], and nearest-neighbor field search [26]. In this work, we show that image self-similarities can help to synthesize more content beyond textures.

A preliminary version of this work has been published in ECCV '12 [27]. The statistics of patch offsets have witnessed other applications beyond image completion. Chen et al. [28] use the dominant offsets to initialize optical flows. They present top-ranked results in optical flow benchmarks. Zhang et al. [29] use the the dominant offsets with the graph-cuts algorithm to extrapolate images. We expect the statistics of patch offsets, as a kind of natural image statistics, will find more applications in the future.

2 Approaches

We first introduce a way of computing the statistics of patch offsets. Based on this, we develop both matching-based and graph-based methods for image completion. We provide analysis in the next section.

2.1 Computing the Statistics of Patch Offsets

To compute the statistics, we first match similar patches *in the known region* and obtain their offsets (Fig. 1b). For each patch P in the known region, we find another known patch that is the most similar with P and compute their relative position $\mathbf{s}$. Formally, the offset $\mathbf{s}$ is:

$$\mathbf{s}(\mathbf{x}) = \arg \min_{\mathbf{s}} \|P(\mathbf{x}+\mathbf{s}) - P(\mathbf{x})\|^2 \quad s.t. \quad |\mathbf{s}| > \tau. \tag{1}$$

Here, $\mathbf{s} = (u, v)$ is the 2d coordinates of the offset, $\mathbf{x} = (x, y)$ is the position of a patch, and $P(\mathbf{x})$ is a $w \times w$ patch centered at $\mathbf{x}$. We represent each patch as a $3w^2$-dimensional vector of its RGB colors. The similarity between two patches is measured by the squared Euclidean distance between their representations. The threshold τ is to preclude nearby patches. This constraint is to avoid trivial statistics as we will discuss.

The solution to Eq. (1) can be approximately obtained by nearest-neighbor field algorithms like the PatchMatch [18] or its improvements [30], [31]. Because we will compute the statistics, the approximation in these algorithms almost does not impact the dominant offsets. In this paper we adopt [31] due to its fast speed.

Given all the offsets $\mathbf{s}(\mathbf{x})$ for all the known pixels $\mathbf{x}$, we compute their statistics by a 2d histogram $h(u, v)$:

$$h(u,v) = \sum_{\mathbf{x}} \delta(\mathbf{s}(\mathbf{x}) = (u,v)), \tag{2}$$

where $\delta(\cdot)$ is 1 when the argument is true and 0 otherwise. We pick out the K highest peaks of this histogram. They correspond to the first K dominant offsets. We empirically use $K = 60$ throughout this paper.

Fig. 1c shows an example of this histogram. There are two major peaks in the horizontal direction. The offsets given by these two peaks indicates how the patterns are mostly repeated in the image. In Section 3 we will discuss various cases and explain how these dominant offsets influence the image completion algorithms.

2.2 Image Completion Using the Statistics of Patch Offsets

As discussed in the introduction, both matching-based and graph-based methods can be viewed as assigning an offsets $\mathbf{s}(\mathbf{x})$ to each unknown pixel/patch at $\mathbf{x}$. The algorithms will copy content from the location $\mathbf{x} + \mathbf{s}$ and paste it (pixel-wise or patch-wise) into the position $\mathbf{x}$. Next we show how the statistics of patch offsets can be applied to both matching-based and graph-based methods.

2.2.1 Graph-Based Image Completion Using the Statistics of Patch Offsets

In our graph-based solution, we treat image completion as a *photomontage* [32] problem. Given the K dominant offsets, we combine a stack of shifted images corresponding to these offsets (see Fig. 1d). Formally, we optimize the following MRF energy function:

$$E(L) = \sum_{\mathbf{x}\in\Omega} E_d(L(\mathbf{x})) + \sum_{(\mathbf{x},\mathbf{x}')|\mathbf{x}\in\Omega,\mathbf{x}'\in\Omega} E_s(L(\mathbf{x}), L(\mathbf{x}')). \tag{3}$$

Here Ω is the unknown region (expanded by one pixel to include boundary conditions). The neighboring pixels $(\mathbf{x}, \mathbf{x}')$ are four-connected. The argument L is a *labeling* map. It assigns a label to each unknown pixel $\mathbf{x}$, where the labels represent the pre-selected offsets $\{\mathbf{s}_i\}_{i=1}^K$ or $\mathbf{s}_0 = (0,0)$. Here $\mathbf{s}_0$ is valid if and only if $\mathbf{x}$ is on the boundary of Ω, so as to impose boundary constraints. Intuitively, "$L(\mathbf{x}) = i$" means that we copy the pixel at $\mathbf{x} + \mathbf{s}_i$ to the location $\mathbf{x}$.

The data term E_d is 0 if the label is valid for $\mathbf{x}$, i.e., $\mathbf{x} + \mathbf{s}$ is a known pixel; otherwise E_d is $+\infty$. The smoothness term E_s penalizes the incoherent seams. Denoting $a = L(\mathbf{x})$ and $b = L(\mathbf{x}')$, we define E_s as:

$$\begin{aligned} E_s(a,b) = &\|I(\mathbf{x}+\mathbf{s}_a) - I(\mathbf{x}+\mathbf{s}_b)\|^2 \\ &+ \|I(\mathbf{x}'+\mathbf{s}_a) - I(\mathbf{x}'+\mathbf{s}_b)\|^2. \end{aligned} \tag{4}$$

Here $I(\mathbf{x})$ is the RGB color of $\mathbf{x}$. Note $I_a(\cdot) \triangleq I(\cdot + \mathbf{s}_a)$ is an image shifted by $\mathbf{s}_a$ (Fig. 1d); and likewise $I_b(\cdot) \triangleq I(\cdot + \mathbf{s}_b)$. If $\mathbf{s}_a \neq \mathbf{s}_b$, the neighboring pixels $\mathbf{x}$ and $\mathbf{x}'$ will be assigned different offsets, i.e., $L(\mathbf{x}) \neq L(\mathbf{x}')$, so there will exist a seam between $\mathbf{x}$ and $\mathbf{x}'$. In this sense, Eq. (4) penalizes the neighboring labels if the two shifted images $I(\cdot + \mathbf{s}_a)$ and $I(\cdot + \mathbf{s}_b)$ are not similar near this seam. This smoothness term is similar to the those used in the GraphCuts texture [33], Photomontage [32], or Shift-map [17] methods.

As in the photomontage problem [32], we optimize the energy (3) using multi-label graph-cuts [21] and then further suppress the seams by Poisson blending [34]. More implementation details are in Section 4. Fig. 1e shows an image completion result of our graph-based solution.

2.2.2 Matching-Based Image Completion Using the Statistics of Patch Offsets

The statistics of patch offsets can also be incorporated into the coherence measure d_{cohere} proposed in [11], [13]. In this paper we adopt the definition used in [14], [18]:

$$d_{\text{cohere}} = \sum_{P\in\Omega} \min_{Q\in\overline{\Omega}} \|P - Q\|^2, \tag{5}$$

where P is a patch in the synthesized region Ω, and Q is a patch in the known region $\overline{\Omega}$. This measure penalizes any patch P in the synthesized region if its best match Q in the known region is not similar to it. In our current implementation we do not apply the weights used in [13].

We can rewrite Eq. (5) in a way of offset assignments:

$$d_{\text{cohere}} = \sum_{\mathbf{x}\in\Omega} \min_{\mathbf{s};\mathbf{x}+\mathbf{s}\in\overline{\Omega}} \|P(\mathbf{x}) - P(\mathbf{x}+\mathbf{s})\|^2, \tag{6}$$

where $P(\mathbf{x} + \mathbf{s})$ is a known patch. Eq. (6) clearly shows that each patch in the Ω will be assigned an offset $\mathbf{s}$. It also implies that there is no constraint on the selection of $\mathbf{s}$: it accepts all valid $\mathbf{s}$ such that $\mathbf{x} + \mathbf{s} \in \overline{\Omega}$.

Based on Eq. (6), it is easy to incorporate the statistics of patch offsets into the coherence measure:

$$\hat{d}_{\text{cohere}} = \sum_{\mathbf{x}\in\Omega} \min_{1\le i\le K} \|P(\mathbf{x}) - P(\mathbf{x}+\mathbf{s}_i)\|^2. \tag{7}$$

This equation means that the offsets can only be chosen from the dominant offsets $\{\mathbf{s}_i\}_{i=1}^K$ obtained from the statistics. Note the patch size here need not be the same as the one used for computing the statistics. We denote this patch size as $w' \times w'$.

The coherence measure in Eq. (7) can be optimized in an EM fashion just as in [14], [18]. In the E-step, the algorithm computes a nearest neighbor field (each unknown pixel assigned an offset) that maps a patch $P \in \Omega$ to its most similar patch Q. This was done by the PatchMatch algorithm in [18]. In contrast, here we simply exhaust $i \in [1, K]$ for each $P(\mathbf{x})$ to find the most similar patch $P(\mathbf{x} + \mathbf{s}_i)$. In the M-step, the color of each unknown pixel is reconstructed by "voting" as in [18]: because each unknown pixel is covered by multiple overlapping patches, all the corresponding pixels in these patches are averaged to give the new color of this pixel. This algorithm is iterated.

As in [14], [18], we also adopt a multi-scale strategy. More implementation details are in Section 4.

2.2.3 Discussions

We have shown the statistics of patch offsets can be naturally incorporated into both graph-based and matching-based methods. In Section 5 we show the results of our both methods. Between the two methods using the statistics, we recommend the graph-based one. Our graph-based solution

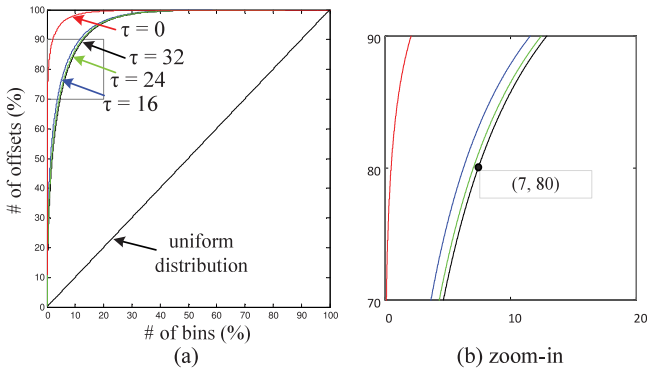


Fig. 2. (a): Cumulative distributions of offsets, averaged over 5,000 images. (b): zoom-in of (a).

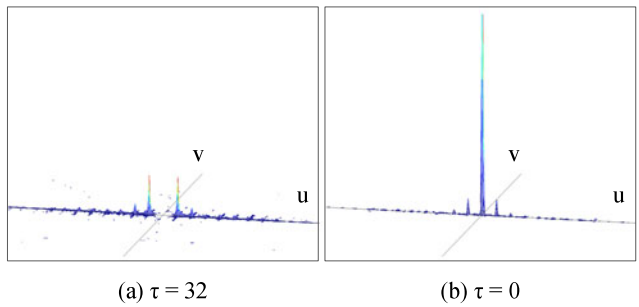


Fig. 3. Offset statistics of Fig. 1 using $\tau = 32$ or $\tau = 0$ (shown in the same scale).

does not require patch representations in the optimization step, so it is less involved in the issue of choosing patch sizes (see Section 3.4). Besides, we find our graph-based solution is faster (see Section 5) thanks to the efficiency of graph-cuts [21]. Unless specified, the results in these paper are obtained from our graph-based solution.

3 Analysis

In this section, we analyze the statistics of patch offsets and their impacts to image completion.

3.1 Sparsity of the Offsets Statistics

One of our key observations is that the offsets statistics are sparse (Fig. 1c). We verify this observation in the MSRA Salient Object Database [35] which contains 5,000 images with manually labeled salient objects. We omit these objects and compute the offsets statistics in the background. Note the background still contains various structures or less salient objects. We use 8×8 patches and test $\tau = 16$, 24, or 32 in Eq. (1). For each image, we sort the histogram bins in the descending order of their magnitude (i.e., the number of offsets in a bin), and we cumulate the bins. The cumulative distribution, averaged over 5,000 images, is shown in Fig. 2a. We can see the offsets are sparsely distributed: (e.g.), when $\tau = 32$, about 80 percent of the offsets are in 7 percent of all possible bins (Fig. 2b). We also observe the cumulative distribution changes only a little with different τ values (16, 24, or 32) (Figs. 2a and 2b). This means the sparsity is insensitive to τ in a wide spectrum.

It is worth mentioning that the non-nearby constraint ($|\mathbf{s}| > \tau$) in Eq. (1) is important. A recent work on natural image statistics [36] shows that the best match of a patch is most probably located near itself. We verify this by setting $\tau = 0$ (thus a patch can match any other patch rather than itself). We can see that the offsets statistics have a single dominant peak around $(0, 0)$ (e.g.), Fig. 3b). Although the offsets distribution is even sparser (see Fig. 2a and Figs. 3a and 3b), the zero offset is insignificant for inferring structures in holes.

3.2 Offsets Statistics for Image Completion

We further observe that the dominant offsets (with the non-nearby constraint) are informative for filling the hole under at least three situations: (i) linear structures, (ii) regular/random textures, and (iii) repeated objects.

3.2.1 Linear Structures

As illustrated in Fig. 4a, a patch on a linear structure can find its match that is also on this structure. The offsets statistics will exhibit a series of peaks along the direction of the structure (Fig. 4b). Here we use a dot to denote a dominant offset in the histogram. These offsets will shift the image along the linear structure, so can reliably complete the mission region.

Fig. 5 shows some real examples of this case. We find our method works well for the linear structures including long and thin objects (Fig. 5a), color edges (Fig. 5b), and textural edges (Figs. 5c and 5d). Note that our method is tolerant to the structures that are not salient (e.g., Fig. 5c) or that are not strictly straight (e.g., Fig. 5d); it is sufficient if the structures have a "trend" along one or a few directions.

3.2.2 Textures

Textures can yield prominent patterns in the offset statistics. Ideally, a *regular* texture should generate a regular pattern of dominant offsets describing how the textures are repeated (Fig. 6). We can complete the texture by shifting the image using these offsets. Fig. 7 shows a real example of regular textures. Because the "period" of the regular textures can be larger than some predefined patch sizes, completing such textures is a challenging task for other techniques like Content-Aware Fill [19] (Fig. 7d). For *irregular* textures, we find the dominant offsets generate random patterns (Fig. 8). In this case our method behaves just like the Graphcut texture algorithm [33].

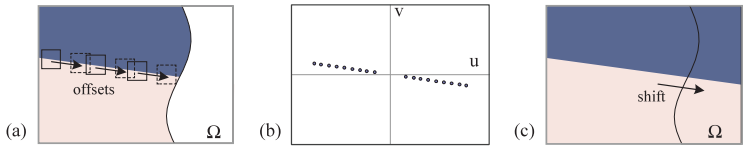


Fig. 4. Illustration of completing linear structures. (a) matching patches. (b) ideal dominant offsets. (c) filling the hole by shifting the image using these offsets. This figure is for illustration only. The real cases are in Fig. 5.

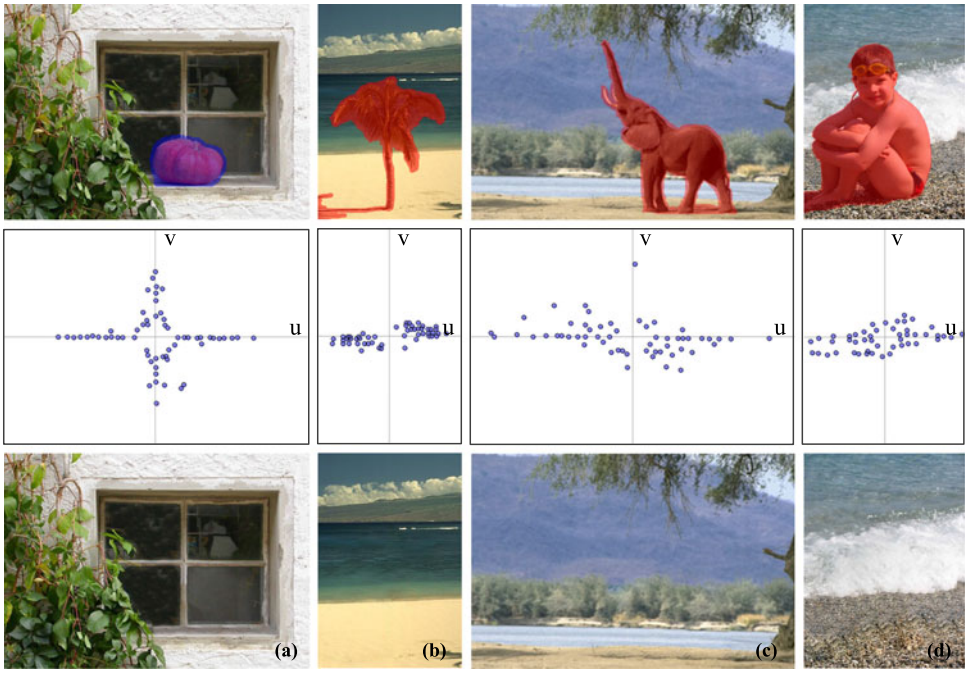


Fig. 5. Linear structures. Top: input. Middle: dominant offsets. Bottom: the results obtained by our graph-based solution. (a) Long/thin objects. (b) Linear color edges. (c), (d) Linear textural edges.

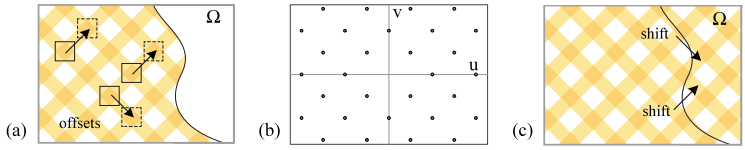


Fig. 6. Illustration of completing textures. The notations are as in Fig. 4.

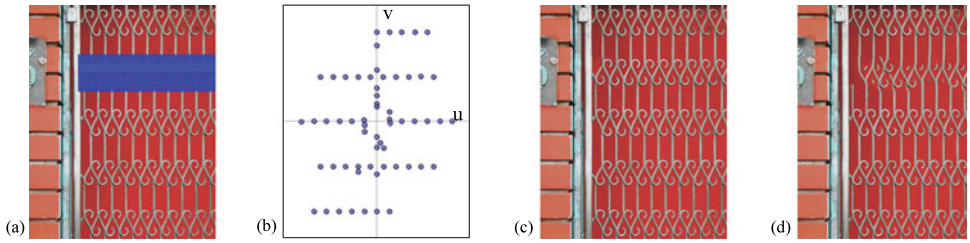


Fig. 7. Regular textures. (a) input. (b) dominant offsets. They describe how the textures are repeated. (c) the result obtained by our graph-based solution. (d) result of Content-Aware Fill.

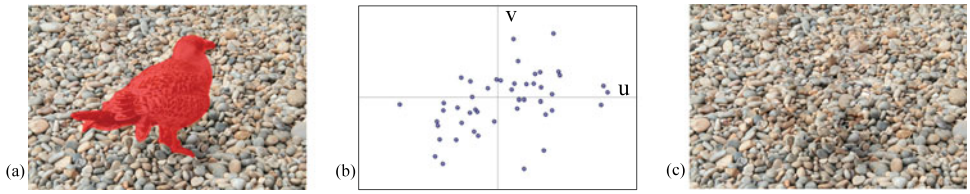


Fig. 8. Random textures. (a) input. (b) dominant offsets. (c) the result obtained by our graph-based solution.

3.2.3 *Repeated Objects*

Repeated objects can also generate prominent peaks in the offset statistics. This is helpful in synthesizing semantic content. As shown in Fig. 9, the partially missing circles yield peaks in the offsets that correspond to the relative positions of the circles. We can complete each circle by shifting another circle with these offsets. In Fig. 10 we show a real example in which our algorithm faithfully recovers a fully missing sculpture. We find that Content-Aware Fill might produce unsatisfactory results (see Figs. 9e and 10f), mainly because it is unaware of how the objects are repeated.

In sum, the offsets statistics can predict the structures in the missing regions in the cases of linear structures, textures, and repeated objects. Our method can handle all of them in the same framework. In the literature, Sun et al.'s work [12] and the PatchMatch paper [18] have proposed to preserve linear structures under the guidance of user

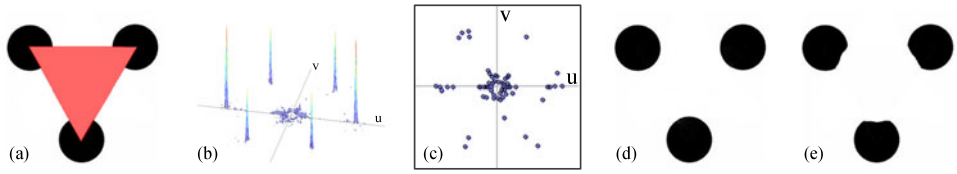


Fig. 9. Repeated circles. (a) input with a missing region in red. (b) offset histogram. (c) dominant offsets found by our algorithm. (d) our result (this is a real result, not a synthetic illustration). (e) result of Content-Aware Fill.

interactions, whereas extensive studies (to name a few, [6], [22], [33]) have been focused on synthesizing texture. Both cases are often encountered in image completion. Our method has the advantages that it need not handle them separately, need not know the cases of line/texture in advance, and needs no user interaction.

3.3 Offsets Selection and the Graph-Based Energy Optimization

Our graph-based method has an energy function Eq. (3) similar to the Shift-map method [17]. The main difference is that Shift-map allows all possible offsets. As a result, our solution space is a very small subset of the one of Shift-map. Theoretically, Shift-map can achieve a smaller energy than our method (this is observed in experiments). However, we find that the results of Shift-map may have unexpected bias, and their visual quality can be unsatisfactory even if their energy is much lower than ours.

In Fig. 11 we optimize the energy Eq. (3) respectively using our selected K dominant offsets (Fig. 11b) and using all possible offsets (Fig. 11c). The later is the way of Shift-map [17] (except that [17] has an extra gradient smoothness term). As expected, our energy (4.6×10^6) is much larger than the energy of Shift-map (1.1×10^6). But our result is visually superior.

We investigate this unexpected phenomenon through the graph-cuts label maps (Figs. 11b and 11c). We find that with a huge number of offsets, the Shift-map method can decrease the energy by "inserting" a great number of insignificant labels into the seam (see the zoom-in of Fig. 11c). These labels correspond to a few isolated pixels that occasionally "connect" the content on both sides of the seam. When the offset candidates are in a great number, these "occasional" pixels are not rare. We further observe that this problem is inherent and cannot be safely avoided by a hierarchical solver [17] (Fig. 11d) or by combining the gradient smoothness term (Fig. 11e, obtained from the authors' demo [37]).

On the contrary, our method is less influenced by this problem (Fig. 11b). Actually, an ideal exemplar-based method should fill the missing region by copying large segments (like patches or regions). This means that only a few offsets should take effect, which is ensured by our method. The above experiments and analysis indicate that reliably limiting the solution space is important for improving the quality in image completion.

Although the dominant offsets selected by our method can improve the quality (and also speed), it is non-trivial to select a few reliable candidate offsets (e.g., 60) out of all possible ones (usually $10^4 - 10^6$). We compare some naive offset selection methods in Fig. 12. We generate the same number ($K = 60$) of offsets, either on a regular grid (Fig. 12b), randomly (Fig. 12c), or by our method (Fig. 12d). We observe that the alternative methods cannot produce satisfactory results, because the predefined offsets do not capture sufficient information to predict the missing structures.

3.4 Patch Sizes for the Offsets Statistics

Most exemplar-based methods (except [17]) involve the issue of setting suitable patch sizes. Our graph-based energy does not rely on patch representations as in [17]; the patch sizes only impact the computation of the offsets statistics (Eq. (1)). As analyzed above, the dominant offsets in the statistics are mainly determined by how the patterns are repeated in the known regions. Such repeatedness is insensitive to the patch sizes.

In Fig. 13 we show two examples using our graph-based solution. Here we test $w \times w = 4 \times 4$, 8×8, 16×16, 24×24, and 32×32 patches used for computing the offsets

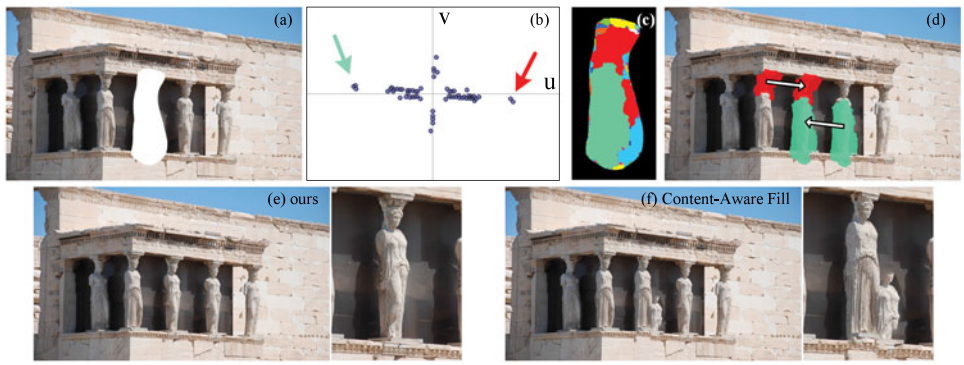


Fig. 10. Repeated objects. (a) input with a sculpture missing. (b) dominant offsets. The arrows indicate the offsets of the relative positions of the sculptures. (c) the label map obtained by graph-cuts: each color represents an offset. (d) the hole is mainly filled by copying the other sculptures, using the offsets indicated in (b). (e) our result and zoom-in. (f) result of Content-Aware Fill and zoom-in.

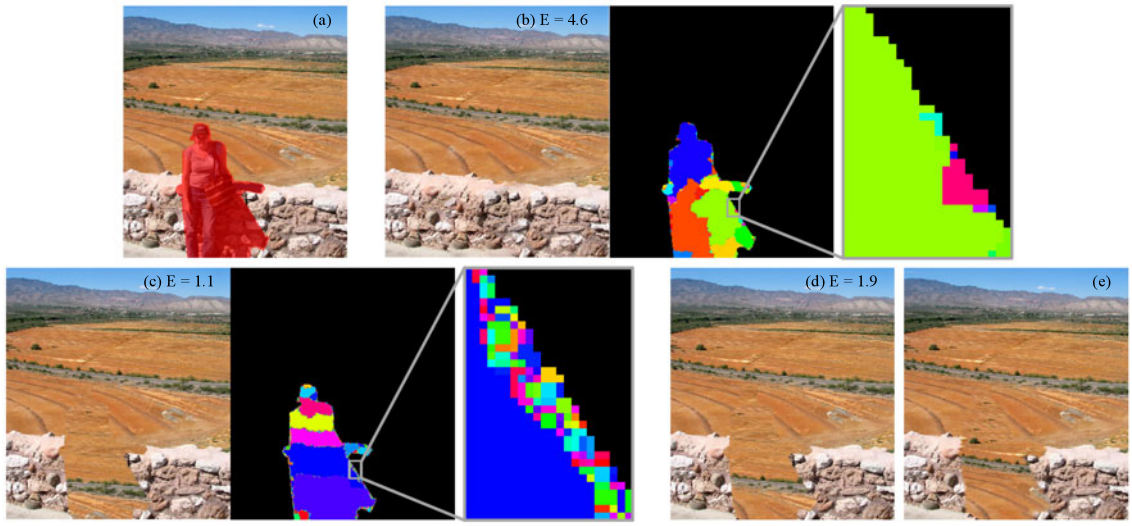


Fig. 11. Offsets sparsity and optimized energy. (a) input. (b) our graph-based result and the label map. Energy: $E = 4.6(\times 10^6)$, running time: $t = 0.5s$. (c) the result using all possible offsets. $E = 1.1(\times 10^6)$, $t = 4{,}300s$. (In this case the number of labels is $K = 3.8 \times 10^5$, and the number of unknown pixels is $N = 2.2 \times 10^4$.) (d) the result of a hierarchical solver [17]. $E = 1.9(\times 10^6)$, $t = 83s$. (e) the result from the authors' demo [37] (with gradient smoothness terms).

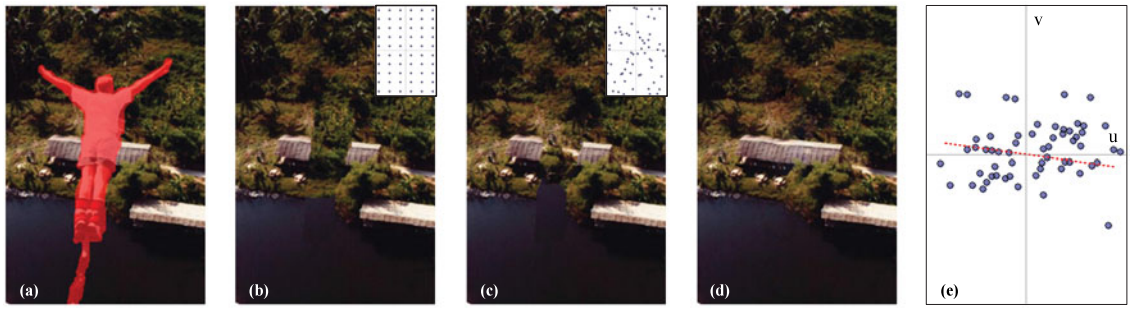


Fig. 12. Comparisons of offsets selection methods. (a) input. (b) the result of regularly spaced offsets. (c) the result of randomly distributed offsets. (d) our result. (e) our offsets. The dash line indicates the offsets used to complete the structure of the roof.

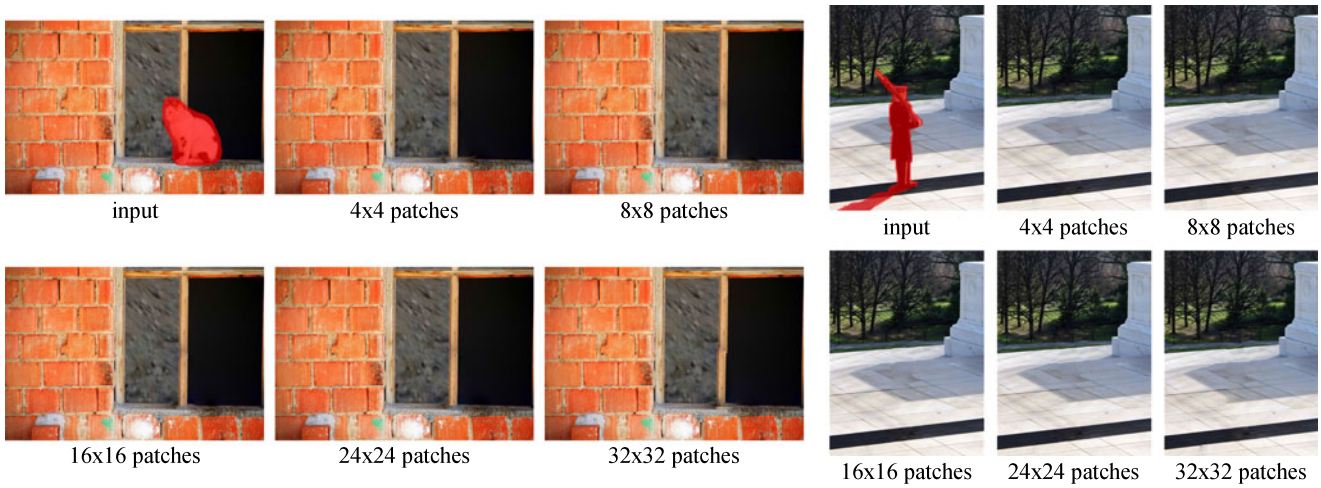


Fig. 13. Image completion results obtained by our graph-based solution, using various patch sizes for computing the statistics of patch offsets.

statistics. We can see that our method can produce visually plausible results in a very side spectrum of patch sizes. This experiment shows that our method is very robust to patch sizes. In all other experiments in this paper, we fix the patch size as 8×8.

4 Implementation Details

4.1 Computing the Statistics

To efficiently matching the patches as in Eq. (1), we apply a nearest-neighbor field algorithm in [31] with a slight

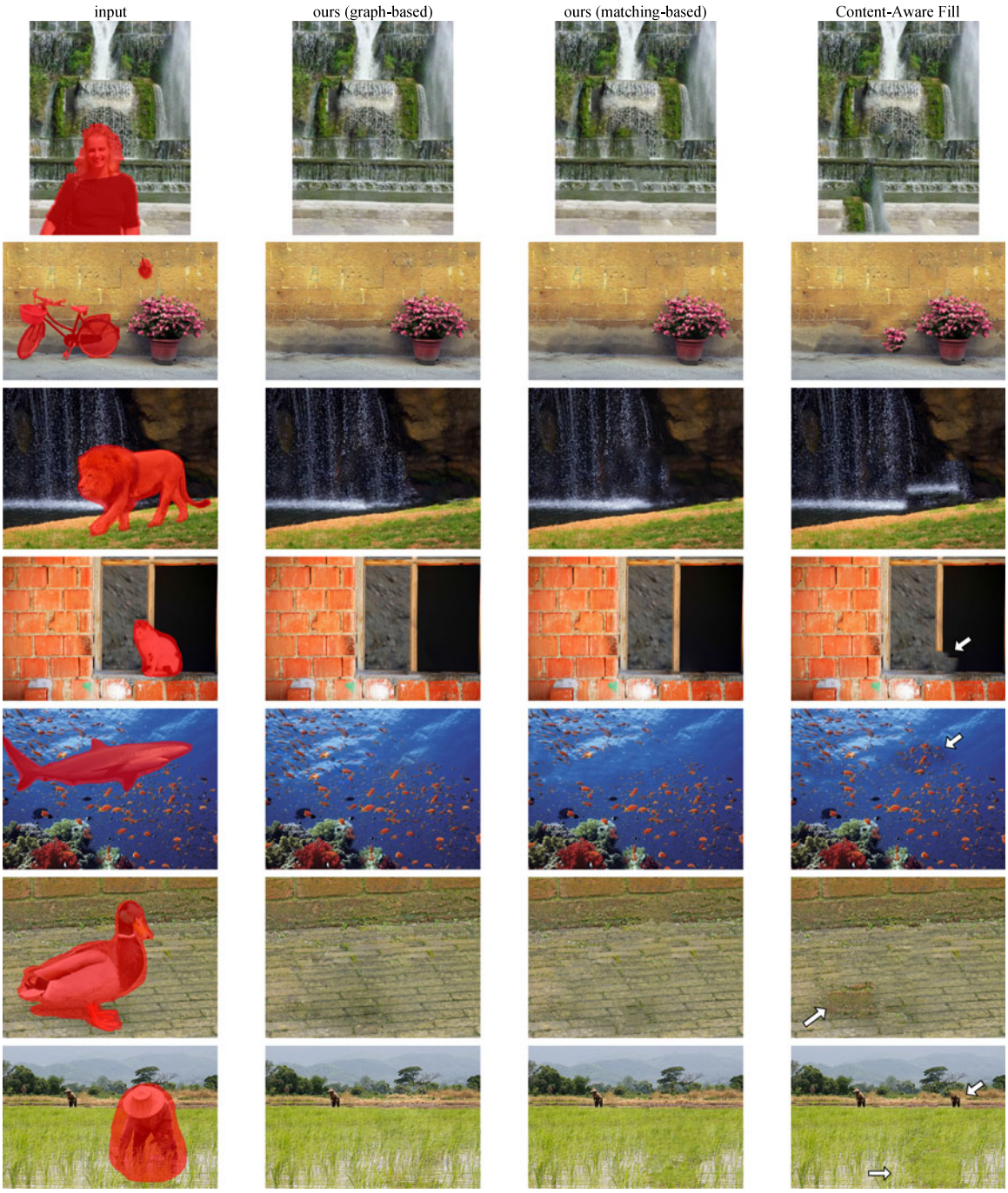


Fig. 14. Comparisons with Content-Aware Fill. From left to right: input, our graph-based results, our matching-based results, and results of Content-Aware Fill. The artifacts are highlighted by the arrows. Image size (from top to down): 0.12, 0.2, 0.26, 0.6, 2, 4, 10 Mp. The running time is in Fig. 15.

modification: to handle the non-nearby constraint ($|\mathbf{s}| > \tau$), before computing the difference between a pair of patches we first check their spatial distance and reject them if the constraint is disobeyed.

We perform this matching step in a rectangular region centered around the bounding box of the hole. This rectangle is three times larger (in length) than this bounding box. The purpose of using such a rectangular region is to avoid unreliable statistics if the hole is too small in practical applications (in most examples in this paper this region is the entire image because the holes are large). The threshold τ in Eq. (1) is set as $\frac{1}{15}\max(\mathrm{w},\mathrm{h})$ where w and h are the width and height of this region. We downsample this region to 800×600 pixels if it is larger than this size. Then we use 8×8 patches to perform the matching step.[1] This step takes $<$0.1 s.

Given the nearest-neighbor field $\mathbf{s}(\mathbf{x})$, we compute the 2d histogram $h(u, v)$ as in Eq. (2). We smooth this histogram by a Gaussian filter ($\sigma = \sqrt{2}$). In this smoothed histogram, we consider a "peak" as a bin whose magnitude is locally maximal inside a 9×9 window. The highest K peaks are picked

1. The method in [31] only supports patch sizes $4k \times 4k$ for some integer k. When computing the offset statistics, we need not use a $(2r+1) \times (2r+1)$ patch that centered at a certain pixel; instead, we can represent the spatial coordinates of a patch by its top-left corner. This representation is also adopted in the public codes of [18] and [30].

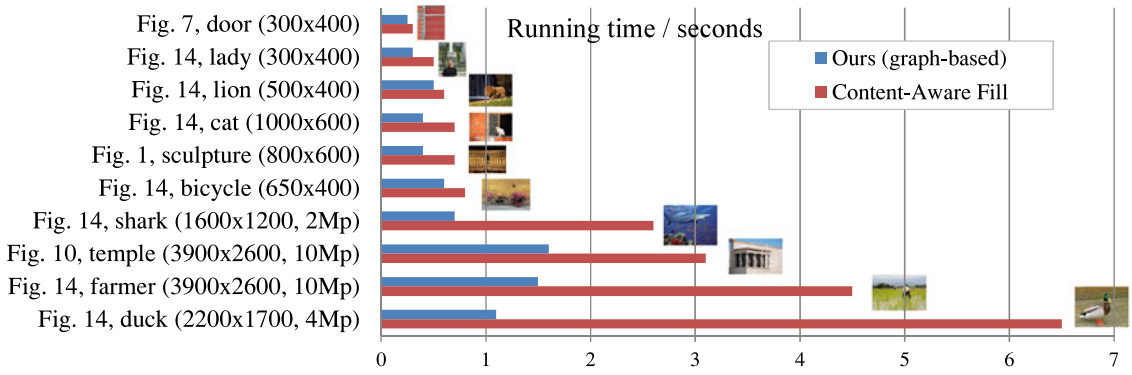


Fig. 15. Running timing comparisons between our graph-based method and Content-Aware Fill. The bars are sorted in the ascending order of Content-Aware Fill's time.

out, and their corresponding offsets give the offset candidates $\{\mathbf{s}_i\}_{i=1}^{K}$ that will be used in the image completion algorithms.

4.2 The Graph-Based Method

We adopt a two-scale solver in our graph-based method (Section 2.2.1). We first downsample the rectangular region (by a scale l) to 800×600 pixels if it is larger than this size. Then we build a graph as in Eq. (3) and optimize it using graph-cuts [21]. We use the public code of the multi-label graph-cuts in [38]. Its time complexity is $O(NK)$, where N is the number of unknown pixels and K is the number of labels. The time of this graph-cuts step is 0.2-0.5 seconds for an 800×600 image with 10-20 percent pixels missing. As a comparison, it takes over one hour to solve such an 800×600 image if all possible offsets are allowed at this scale ($K = 10^4$-10^6, like Fig. 11c), or tens of seconds using a hierarchical solver [17] with the coarsest level as small as 100×100 pixels (like Fig. 11d). Thus our method is 1-2 orders of magnitude faster than Shift-map [17].

We upsample the above resulting label map to the full resolution by nearest-neighbor interpolation and multiply the offsets by l. To correct small misalignments, we optimize a cost similar to Eq. (3) in the full resolution. We allow each pixel to take five offsets: the upsampled offset and 4 relative offsets: if the upsampled shift is $\mathbf{s} = (u, v)$, then the other four shifts are $(u \pm \frac{l}{2}, v)$ and $(u, v \pm \frac{l}{2})$. In this cost function we only treat the pixels as unknowns if they are in $\frac{l}{2}$-pixel around the seams. This upsampling takes <0.1 s for typical 2 Mp images. We have also tested our graph-based method in full resolution without downsampling, and found the visual qualities are similar to the two-scale solver. We adopt the two-scale solver because it is faster.

Finally a Poisson fusion [34] is applied to hide the possibly visible seams. We adopt a recent $O(N)$ time Poisson solver proposed in [39]. In our implementation it takes 50 ms per Megapixel.

4.3 The Matching-based Method

As in [14], [18], we adopt a multi-scale solver in our matching-based method (Section 2.2.2). We build an image pyramid of L levels using a scaling factor 2, with a fixed coarsest size ($\geq 100 \times 100$). We start the EM algorithm from the coarsest level, with an initialization discussed below. The result of a coarser level is interpolated into the next finer level. We interpolate the color image and at the next level start from the E-step. (Alternatively, we can interpolate the nearest neighbor field and at the next level start from the M-step. We find the former way is better at hiding the seams.) We run 20 iterations of EM steps in the coarsest level, two iterations in the finest level, and five iterations in other levels.

The result of the matching-based method is sensitive to the initialization. We have tested two ways of initializing at the coarsest level: using the Poisson equation [34] to roughly propagate colors and generate a smoothed guess, or use our graph-based solution to generate a structural guess. We find the second way is more robust and we report the results using this way.

Unlike the graph-based method, the matching-based method requires to set a patch size in its cost function (7). We fixed this size as $w' \times w' = 9 \times 9$ throughout this paper, although in some cases we find adjusting this size can give better results.

5 Experimental Results

All experiments are run on a PC with an Intel Core i7 3.0 GHz CPU and 8 G RAM. We recommend viewing the supplementary video, which can be found on the Computer

Fig. 16. Some results of Kopf et al.'s method [41]. The input images are in Figs. 1, 5, 7, 11, and 13.

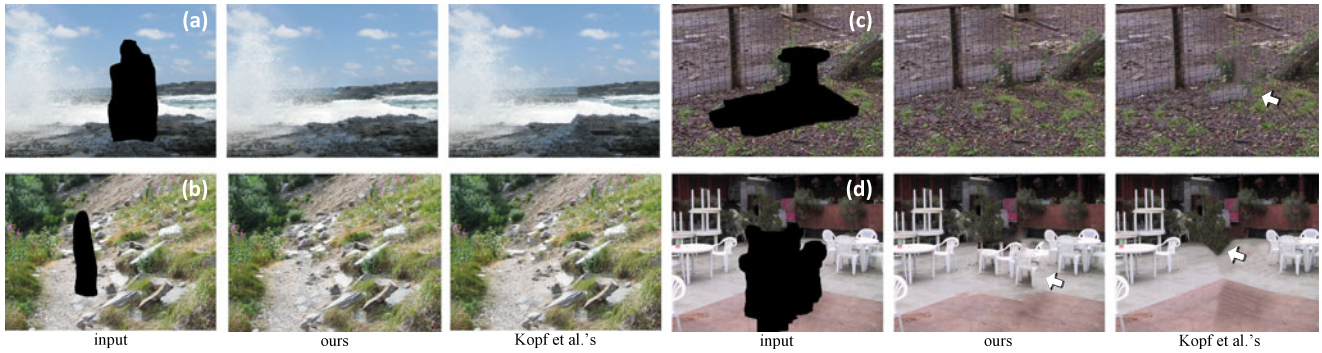


Fig. 17. Comparisons with Kopf et al.'s [41] method on interior holes. Kopf et al.'s results are from their website.

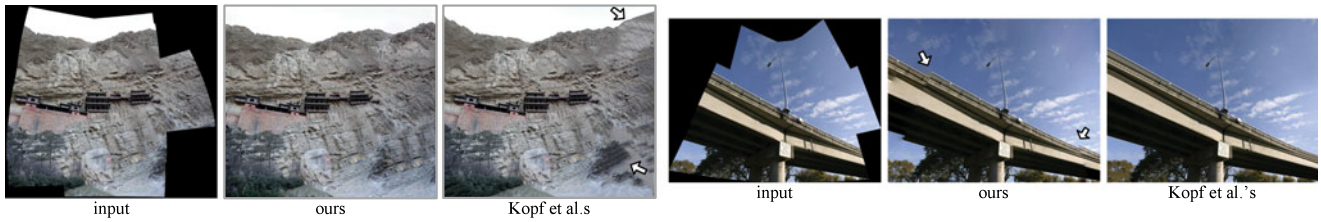


Fig. 18. Comparisons with Kopf et al.'s [41] method on panoramas. Kopf et al.'s results are from their website.

Fig. 19. Comparisons with state-of-the-art methods. (a) Input (640×430). (b) Ours (graph-based, 0.18 s). (c) Content-Aware Fill (0.3 s). (d) Priority-BP [15] (117 s). (e) Shift-map [17] (13 s). (f) Criminisi et al.'s [7] (6.9 s).

Q1 Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2014.2330611, to experience the user interactions and speed.

5.1 Comparisons with Content-Aware Fill

It is widely considered that Wexler's method [11], combined with PatchMatch [18], is one of the state-of-the-art image completion techniques. We have tested a re-implementation of this method using the public code in [40] and our own re-implementation (as described in Section 2.2.2, without using the patch statistics). We find that it is non-trivial to tune the parameters (patch size, pyramid levels, etc.), while the results are sensitive to them. To this end, we choose to compare with the tool Content-Aware Fill in Adobe Photoshop. It has shown compelling quality and speed in many practices. In the official website [19], it reports that Content-Aware Fill in Photoshop CS 5 is based on [11]+[18]. Our experiments are on this version.

Some comparisons have been shown in the previous sections (Figs. 1, 7, 9, and 10). In Fig. 14 we show more examples using our both methods (graph-based and matching-based). In all these examples our methods complete the images using as few as $K = 60$ pre-selected offsets. Our both methods generate high-quality results, whereas the Content-Aware Fill produces noticeable artifacts in these examples.

Fig. 15 shows the running time of our graph-based solution and the Content-Aware Fill. Both methods are using

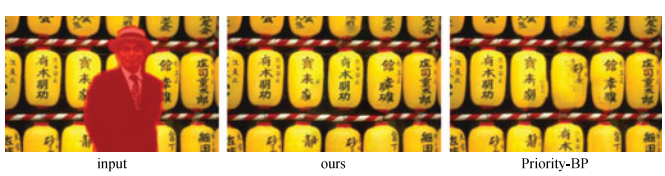


Fig. 20. A comparison with Priority-BP [16]. For this 256 × 163 image Priority-BP takes 40 s in a well parallelized quad-core implementation, while our graph-based method takes 0.09 s.

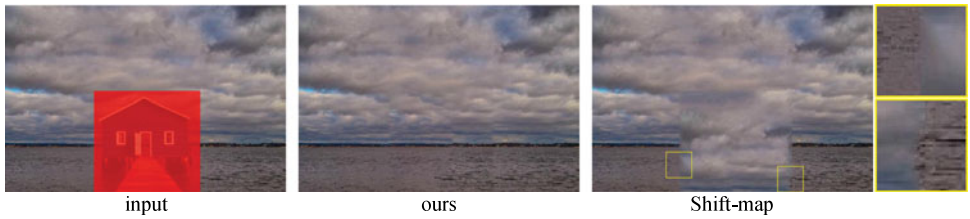


Fig. 21. A comparison with Shift-map [17]. Note how Shift-map behaves near inconsistent seams.

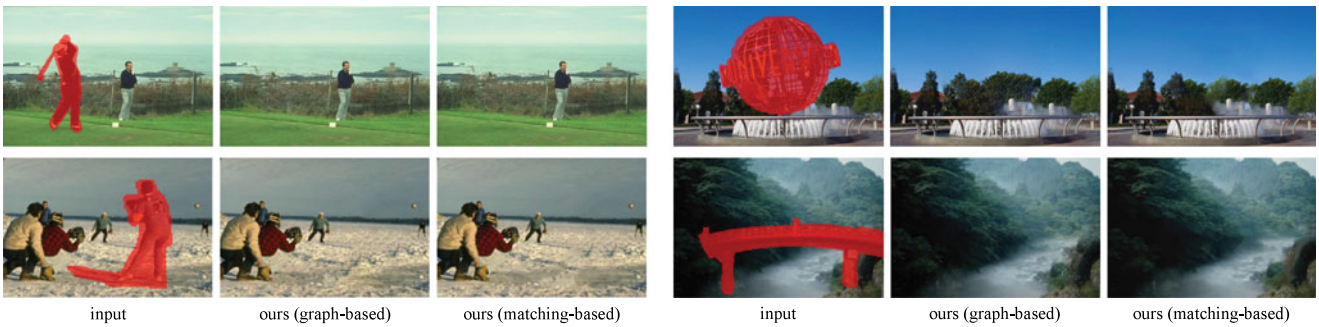


Fig. 22. More results of images from previous papers [8], [13]. These images are around 400×300. Our methods complete each image in less than 0.3 seconds.

quad cores. (Our graph-based method benefits *less* than Content-Aware Fill in multi-core, mainly because in Content-Aware Fill the EM algorithm and PatchMatch are fully parallelized, but the graph-cuts algorithm we used is not. In our graph-based method, the parallelism is only for matching patches and Poisson blending.) For small images where the two-scale solution does not take effect (the first six examples in Fig. 15), our graph-based method is slightly faster than Content-Aware Fill. But for mega-pixel images our graph-based method is two to five times faster. This is because matching patches in mega-pixel images can be slow at finer scales. Our matching-based method takes about 50-100 percent more time than our graph-based method.

5.2 Comparisons with Kopf et al.'s [41]

In [41] Kopf et al. propose to improve Wexler et al.'s method [11] by constraining the patch search region near the hole. We have compared with this method in all 25 images used in our paper and all 50 images provided by [41]. In [41] the 50 images are divided into two sets: interior holes (25 cases) and panoramas (25 cases). All the 75 comparisons are in the supplementary materials, available online.

In Fig. 16 we show some results of Kopf et al.'s method in our successful cases. We can see that Kopf et al.'s method fails synthesizing semantic structures like regular patterns or lines. We also see that Kopf et al.'s method can produce artifacts by tiling patches (see the second image in Fig. 16). This is because when the patch search *positions* (instead of *offsets*) are constrained as in [41], the patch source is limited, and the same patches can be used multiple times.

Fig. 17 shows some results on Kopf et al.'s image set of interior holes. We notice that this image set can be roughly categorized into two subsets: natural scenes (e.g., mountains, grasses, trees; see Figs. 17a and 17b) and very complicated man-made scenes (e.g., street-views, buildings, fences; see Figs. 17c and 17d). The results show that both ours and Kopf et al.'s method can mostly succeed for the natural scenes (Figs. 17a and 17b). On the contrary, the subset of man-made scenes are challenging and both methods often produce artifacts (Fig. 17d). But in case when the man-made scene exhibits regular patterns, our method can potentially produce better results (Fig. 17c).

Fig. 18 shows some results on Kopf et al.'s image set of panoramas. In this set we find that our method is still able to generate acceptable results for simpler natural scenes (Fig. 18 left). But our method is inferior to Kopf et al.'s method for man-made scenes (Fig. 18 right). This is because when "out-painting" the images, the missing pixels can be far away from each other and so have little relationship. Each local region may have its favorable offsets, so our global statistics may fail capturing these offsets. On the contrary, image "out-painting" is particularly suitable for Kopf et al.'s method, because when synthesizing the outside of the image, the good candidate patches can be expected to be located near the missing region. But we should indicate that Kopf et al.'s method still produces obvious artifacts in a large portion of the man-made scenes (see supplements, available online), and cropping is still needed as investigated in their paper.

5.3 Comparisons with Other Methods

In Fig. 19 we further compare with Priority-BP [15], Shift-map [17], and Criminisi et al.'s method [7]. Our method faithfully recovers the texture edges here, and is much faster than the other three methods (see the caption in Fig. 19). More comparisons are in the supplementary materials, available online.[2]

Fig. 20 shows a comparison with Priority-BP [15]. This method optimizes an MRF using the BP algorithm with on-the-fly label pruning. Its running time for this 256×163-pixel image is 40s using a well parallelized quad-core implementation, while our graph-based method takes 0.09 s. Also

2. research.microsoft.com/en-us/um/people/kahe/eccv12.

Fig. 23. Failure of the statistics. (a) Input. (b) Our graph-based result. (c) Result of Content-Aware Fill. (d) An extra stroke is casually painted on the image. (e) Our graph-based result of (d).

note Priority-BP cannot recover the pattern in the lamp in this example.

Fig. 21 shows a comparison with Shift-map [17]. As also in Fig. 11, Shift-map cannot preserve the structure in this case. We can see (in zoom-in) how this method conceals a seam when the content is not consistent on both sides of this seam. This result is obtained from the authors' on-line demo [37]. We tried various parameter settings but observed similar artifacts.

In Fig. 22 we show more results in the example images from previous papers [8], [13].

5.4 Limitations

Fig. 18 (right) has shown that our method is less suitable for "outpainting". Our methods may also fail when the desired offsets do not form dominant statistics. Fig. 23b show a failure case. We can partially solve this problem by manually introducing offsets. (e.g.,) we can paint an extra stroke on the image (Fig. 23d), and treat this image as a new source for patch statistics. This stroke contributes to the statistics and overcomes the problem (Fig. 23e). Some other failure examples are in the supplementary materials, available online.

6 Conclusion

In this paper we have presented novel statistics of patch offsets. We have demonstrated the effects of these statistics for image completion using both graph-based and matching-based methods.

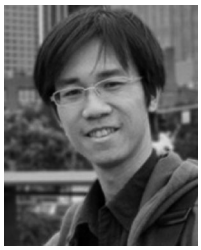
The usage of the patch offsets implies that we only consider shifts of patches for image completion. More complex transforms like scaling, rotation, reflection, and their combinations have been studied, e.g., in Generalized PatchMatch [42], Transforming Image Completion [40], and Image Melding [43]. These transforms are necessary in case when shifting is not sufficient, e.g., completing circles or reflection-symmetric objects). Since these transformations further increase the dimensionality of the search space, it could be useful to investigate the statistics and limit the search (some successful attempts have been shown in a recent work [29] inspired by our method). Image Melding [43] further improves Wexler et al.'s method [11] by voting in the gradient domain. It is possible to combine this method with our matching-based solution.

Our method is based on a kind of natural image statistics. Natural image statistics are essential for many computer vision problems. Gradient-domain statistics have been applied in denoising [44], deconvolution [45], and diffusion-based inpainting [3]. Patch-domain statistics have been shown very successful in denoising [44] and super-resolution [46]. We believe our statistics of patch offsets, as a kind of natural image statistics, will find more applications in the future (e.g., [28], [29]).

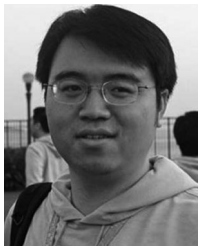
References

[1] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Proc. 27th Annu. Conf. Comput. Graph. Interact. Techn.*, 2000, pp. 417–424.
[2] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera, "Filling-in by joint interpolation of vector fields and gray levels," *IEEE Trans. Image Process.*, vol. 10, no. 8, pp. 1200–1211, Aug. 2001.
[3] A. Levin, A. Zomet, and Y. Weiss, "Learning how to inpaint from global image statistics," in *Proc. 9th IEEE Int. Conf. Comput. Vis.*, 2003, pp. 305–312.
[4] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher, "Simultaneous structure and texture image inpainting," in *Proc. IEEE Comput. Vis. Pattern Recog.*, 2003, pp. 707–712.
[5] S. Roth and M. J. Black, "Fields of experts: A framework for learning image priors," in *Proc. IEEE Comput. Vis. Pattern Recog.*, 2005, pp. 860–867.
[6] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *Proc. IEEE Int. Conf. Comput. Vis.*, 1999, pp. 1033–1038.
[7] A. Criminisi, P. Perez, and K. Toyama, "Object removal by exemplar-based inpainting," in *Proc. IEEE Comput. Vis. Pattern Recog.*, 2003, pp. 721–728.
[8] I. Drori, D. Cohen-Or, and H. Yeshurun, "Fragment-based image completion," in *Proc. Annu. Conf. Comput. Graph. Interact. Techn.*, 2003, pp. 303–312.
[9] J. Jia and C.-K. Tang, "Image repairing: Robust image synthesis by adaptive ND tensor voting," in *Proc. IEEE Comput. Vis. Pattern Recog.*, 2003, vol. 1, pp. I-643–I-650.
[10] J. Jia and C.-K. Tang, "Inference of segmented color and texture description by tensor voting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 6, pp. 771–786, Jun. 2004.
[11] Y. Wexler, E. Shechtman, and M. Irani, "Space-time video completion," in *Proc. IEEE Comput. Vis. Pattern Recog.*, 2004, pp. 120–127.
[12] J. Sun, L. Yuan, J. Jia, and H.-Y. Shum, "Image completion with structure propagation," in *Proc. Annu. Conf. Comput. Graph. Interact. Techn.*, 2005, pp. 861–868.
[13] Y. Wexler, E. Shechtman, and M. Irani, "Space-time completion of video," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 3, pp. 463–476, Mar. 2007.
[14] D. Simakov, Y. Caspi, E. Shechtman, and M. Irani, "Summarizing visual data using bidirectional similarity," in *Proc. IEEE Comput. Vis. Pattern Recog.*, 2008, pp. 1–8.
[15] N. Komodakis and G. Tziritas, "Image completion using global optimization," in *Proc. IEEE Comput. Vis. Pattern Recog.*, 2006, pp. 442–452.
[16] N. Komodakis and G. Tziritas, "Image completion using efficient belief propagation via priority scheduling and dynamic pruning," *IEEE Trans. Image Process.*, vol. 16, no. 11, pp. 2649–2661, Nov. 2007.
[17] Y. Pritch, E. Kav-Venaki, and S. Peleg, "Shift-map image editing," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2009, pp. 151–158.
[18] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "PatchMatch: A randomized correspondence algorithm for structural image editing," in *Proc. Annu. Conf. Comput. Graph. Interact. Techn.*, 2009, pp. 1–8.
[19] Adobe Systems Inc. (2009) [Online]. Available: www.adobe.com/technology/graphics/content_aware_fill.html

[20] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proc. 15th Conf. Uncertainty Artif. Intell.*, 1999, pp. 467–475.
[21] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 11, pp. 1222–1239, Nov. 2001.
[22] V. G. Kim, Y. Lipman, and T. A. Funkhouser, "Symmetry-guided texture synthesis and manipulation," *ACM Trans. Graph.*, vol. 31, no. 3, p. 22, 2012.
[23] E. Shechtman and M. Irani, "Matching local self-similarities across images and videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2007, pp. 1–5.
[24] T. Deselaers and V. Ferrari, "Global and efficient self-similarity for object classification and detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2010, pp. 1633–1640.
[25] S. Bagon, O. Boiman, and M. Irani, "What is a good image segment? A unified approach to segment extraction," in *Proc. 10th Eur. Conf. Comput. Vis.*, 2008, pp. 30–44.
[26] C. Barnes, "Patchmatch: A fast randomized matching algorithm with application to image and video," PhD dissertation, Department of Computer Science, Princeton University, Princeton, NJ, USA, 2011.
[27] K. He and J. Sun, "Statistics of patch offsets for image completion," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 16–29.
[28] Z. Chen, H. Jin, Z. Lin, S. Cohen, and Y. Wu, "Large displacement optical flow from nearest neighbor fields," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2013, pp. 2443–2450.
[29] Y. Zhang, J. Xiao, J. Hays, and P. Tan, "Framebreak: Dramatic image extrapolation by guided shift-maps," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2013, pp. 1171–1178.
[30] S. Korman and S. Avidan, "Coherency sensitive hashing," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 1607–1614.
[31] K. He and J. Sun, "Computing nearest-neighbor fields via propagation-assisted KD-trees," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2012, pp. 111–118.
[32] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen, "Interactive digital photomontage," in *Proc. Annu. Conf. Comput. Graph. Interact. Techn.*, 2004, pp. 294–302.
[33] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut textures: Image and video synthesis using graph cuts," in *Proc. Annu. Conf. Comput. Graph. Interact. Techn.*, 2003, pp. 277–286.
[34] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," in *Proc. Annu. Conf. Comput. Graph. Interact. Techn.*, 2003, pp. 313–318.
[35] T. Liu, J. Sun, N.-N. Zheng, X. Tang, and H.-Y. Shum, "Learning to detect a salient object," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2007, pp. 1–8.
[36] M. Zontak and M. Irani, "Internal statistics of a single natural image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2011, pp. 977–984.
[37] Shift-map On-line Demo. (2009). [Online]. Available: www.vision.huji.ac.il/shiftmap/
[38] Multi-label Graph Cuts. (2010). [Online]. Available: http://vision.csd.uwo.ca/code/
[39] Z. Farbman, R. Fattal, and D. Lischinski, "Convolution pyramids," in *Proc. SIGGRAPH Asia Conf.*, 2011, p. 175.
[40] A. Mansfield, M. Prasad, C. Rother, T. Sharp, P. Kohli, and L. V. Gool, "Transforming image completion," in *Proc. British Mach. Vis. Conf.*, Aug. 2011, pp. 121.1–121.11.
[41] J. Kopf, W. Kienzle, S. Drucker, and S. B. Kang, "Quality prediction for image completion," in *Proc. SIGGRAPH Asia Conf.*, 2012, pp. 131:1–131:8.
[42] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein, "The generalized patchmatch correspondence algorithm," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 29–43.
[43] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen, "Image melding: Combining Inconsistent images using patch-based synthesis," in *Proc. Annu. Conf. Comput. Graph. Interact. Techn.*, 2012, pp. 82:1–82:10.
[44] V. Katkovnik, A. Foi, K. Egiazarian, and J. Astola, "From local kernel to nonlocal multiple-model image denoising," *Int. J. Comput. Vis.*, vol. 86, no. 1, pp. 1–32, 2010.
[45] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman, "Removing camera shake from a single photograph," in *Proc. Annu. Conf. Comput. Graph. Interact. Techn.*, 2006, pp. 787–794.
[46] D. Glasner, S. Bagon, and M. Irani, "Super-resolution from a single image," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2009, pp. 349–356.



Kaiming He received the BS degree from Tsinghua University in 2007, and the PhD degree from the Chinese University of Hong Kong in 2011. He is a researcher at Microsoft Research Asia (MSRA). He joined Microsoft Research Asia in 2011. His research interests include computer vision and computer graphics. He has won the Best Paper Award at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2009. He is a member of the IEEE.



Jian Sun received the BS degree, MS degree, and PhD degree from Xian Jiaotong University in 1997, 2000, and 2003. He joined Microsoft Research Asia in July, 2003. His research is in the fields of computer vision and computer graphics, with particular interests in Interactive Compute Vision (user interface + vision), and Internet Compute Vision (large image collection + vision). He is also interested in stereo matching, computational photography, face recognition, and deep learning. He has won the Best Paper Award at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2009. He is a member of the IEEE.