Statistics of Patch Offsets for Image Completion

Kaiming He and Jian Sun

Microsoft Research Asia

Abstract. Image completion involves filling missing parts in images. In this paper we address this problem through the statistics of patch offsets. 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. With these offsets we fill the missing region by combining a stack of shifted images via optimization. A variety of experiments show that our method yields generally better results and is faster than existing state-of-the-art methods.

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 must be visually plausible; on the other hand, the algorithm must be efficient, because in practice image completion 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 [1–5]. These methods are mainly designed for filling narrow or small holes (known as "inpainting") but work less well for large missing regions because they are less suitable for synthesizing semantic textures or structures.

Another category of methods that are more effective for large holes are exemplar-based. We further categorize them into two groups: *matching-based* and *MRF-based*. Matching-based methods [6–10] explicitly match patches in the region to be filled with those in the known region and copy the content. This makes it possible to synthesize textures [6] and more complex content [7–10]. The methods in [6–8] progress in greedy fashions, whereas [9] adopts a global cost function called *coherence measure*, where each patch in the filled region is optimally similar to a known patch. This measure helps to yield more coherent results and is later generalized for image retargeting/reshuffling [11]. But because the underlying cost has multiple disconnected local optima, this method is very sensitive to initialization and to the optimization strategy.

Matching-based methods are computationally expensive. A fast PatchMatch method [12] largely relieves this problem and is combined with [9]. The *Content Aware Fill* in Adobe Photoshop, implementing [9, 12] as reported in [13], is arguably the current state-of-the-art image completion software in terms of quality and speed.

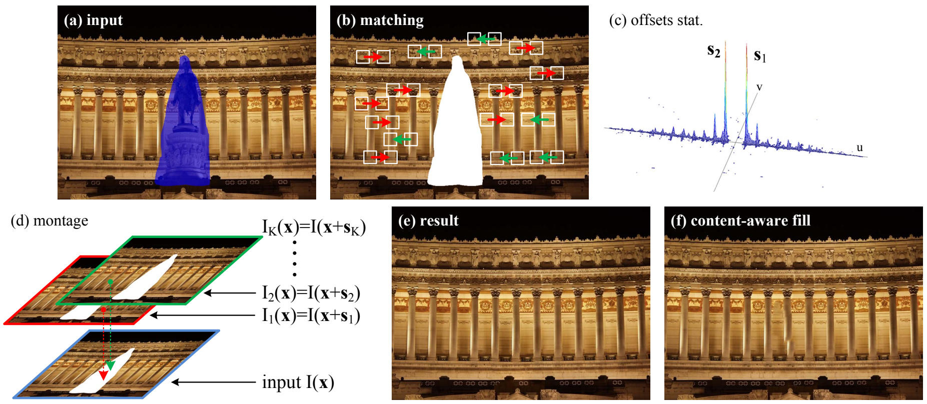
(a) input
(b) matching
(c) offsets stat.
s_2
s_1
v
u
(d) montage
$I_K(\mathbf{x})=I(\mathbf{x}+\mathbf{s}_K)$
$I_2(\mathbf{x})=I(\mathbf{x}+\mathbf{s}_2)$
$I_1(\mathbf{x})=I(\mathbf{x}+\mathbf{s}_1)$
input $I(\mathbf{x})$
(e) result
(f) content-aware fill

Fig. 1. Algorithm 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 corresponding to the highest peaks are picked out. (d) Combining a set of shifted images with the given offsets. (e) Our result. (f) Result of Content-Aware Fill.

Exemplar-based methods can also be realized by optimizing discrete Markov Random Fields (MRFs), as in Priority-BP [14] and Shift-map [15]. Rather than match patches, these methods rearrange the patch/pixel locations to complete the image. The labels to be optimized in the MRFs are absolute locations [14] or relative offsets [15]. The MRFs are optimized via belief propagation (BP) [14] or graph-cuts [15]. Though avoiding matching patches, these methods are computationally expensive: the time complexity is linear in the number of labels and also in the number of unknown pixels, so is quadratic in the image pixel number. Even with the help of a hierarchical solver [15] or label pruning [14], they may still take tens of seconds to process small images (*e.g.*, 400×300).

We notice that exemplar-based methods, either matching-based or MRF-based, must implicitly or explicitly assign each unknown pixel/patch an *offset* - the relative location from where it copies the content. But existing methods do not predict reliable offsets beforehand, and generally allow any possible offsets. We will show this is not beneficial. In terms of quality, unrestricted labels may lead to unexpected bias for MRF optimization (*cf.* Sec. 3.3), or make the coherence measure [9] difficult to be optimized (*cf.* Sec. 3.4). In terms of speed, a huge label set demands very long running time for MRF-based methods.

In this paper we propose a novel method 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 (as in Fig. 1(c)): 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 clues for completing the missing region. We observe that these offsets are predictive for completing linear structures, textures, and repeated objects. With a few (*e.g.*, 60) pre-selected dominant offsets, we combine a stack of shifted images in the missing region via optimization. A variety of experiments show our method performs generally better and is faster than many state-of-the-art methods, including the Content-Aware Fill in Photoshop.

2 Algorithm

In this section we describe the pipeline of our algorithm. We provide reasoning in the next section. Our method has three steps: (i) matching similar patches; (ii) computing offsets statistics; (iii) filling the hole by combining shifted images.

(i) Matching Similar Patches. We first match similar patches *in the known region* and obtain their offsets (Fig. 1(b)). Formally, for each patch P in the known region, we compute its offset $\mathbf{s}$ to its most similar patch:

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

Here, $\mathbf{s} = (u, v)$ is the 2-d coordinates of the offset, $\mathbf{x} = (x, y)$ is the position of a patch, and $P(\mathbf{x})$ is a patch centered at $\mathbf{x}$. The similarity is measured by sum of squared differences between two patches. The threshold τ is to preclude nearby patches. This constraint is to avoid trivial statistics as we will soon discuss.

(ii) Computing Offset Statistics. Given all the offsets $\mathbf{s}(\mathbf{x})$, we compute their statistics by a 2-d histogram $h(u, v)$:

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

where $\delta(\cdot)$ is 1 when the argument is true and 0 otherwise. We pick out the K highest peaks of this histogram (details in Sec. 4). They correspond to K dominant offsets. We empirically use $K = 60$ throughout this paper.

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

(iii) Combining Shifted Images via Optimization. Given the K offsets, we treat image completion as a *photomontage* [16] problem (Fig. 1(d)): the missing region is filled by combining a stack of shifted images. Formally, we optimize the following MRF energy function:

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

Here Ω is the unknown region (with boundary conditions), and $(\mathbf{x}, \mathbf{x}')$ are 4-connected neighbors. L is the *labeling* where the labels represent the pre-selected offsets $\{\mathbf{s}_i\}_{i=1}^{K}$ or $\mathbf{s}_0 = (0, 0)$[1]. Intuitively, "$L(\mathbf{x}) = i$" means that we copy the pixel at $\mathbf{x} + \mathbf{s}_i$ to the location $\mathbf{x}$.

[1] Here $\mathbf{s}_0$ happens if and only if on boundary pixels

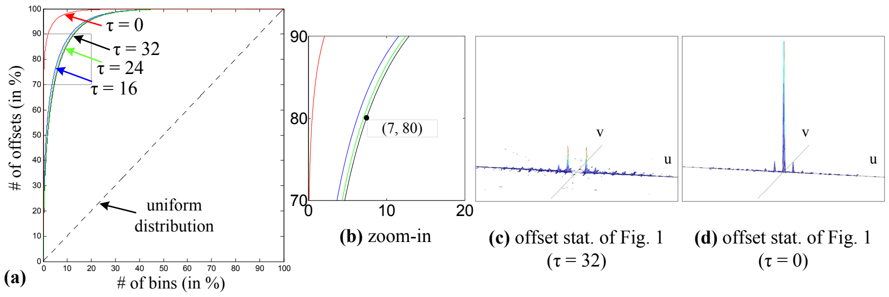

Fig. 2. (a): cumulative distributions of offsets, averaged over 5,000 images. (b): zoon-in of (a). (c)(d): offset statistics of Fig. 1 using τ=32 or τ=0 (shown in the same scale).

The data term E_d is 0 if the label is valid (*i.e.*, $\mathbf{x}+\mathbf{s}$ is a known pixel); otherwise it 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:

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

Here $I(\mathbf{x})$ is the RGB color of $\mathbf{x}$. Note $I(\cdot+\mathbf{s})$ is a shifted image given fixed $\mathbf{s}$. If $\mathbf{s}_a \neq \mathbf{s}_b$, there is a seam between $\mathbf{x}$ and $\mathbf{x}'$. Thus (4) penalizes such a seam that 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 essentially similar to the those defined in GraphCuts texture [17], Photomontage [16], or Shift-map [15]. We optimize the energy (3) using multi-label graph-cuts [18]. Fig. 1(e) shows a result.

3 Analysis

Given the algorithm described above, we have the following observations.

3.1 Sparsity of Offsets Statistics

One of our key observations is that the offsets statistics are sparse (Fig. 1(c)). We verify this in the *MSRA Salient Object Database* [19] 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 τ=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. 2(a). We can see the offsets are sparsely distributed: *e.g.*, when τ=32, about 80% of the offsets are in 7% of all possible bins. We also observe the cumulative distribution changes only a little with different τ values (16, 24, or 32) (Fig. 2(a)(b)). This means the sparsity is insensitive to τ in a wide spectrum.

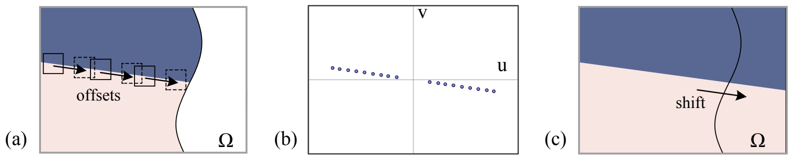

Fig. 3. Illustration of completing linear structures. (a): matching similar patches. (b): ideal dominant offsets. (c) filling the hole by shifting the image using these offsets. Note this figure is for illustration only. The real examples are in Fig. 4.

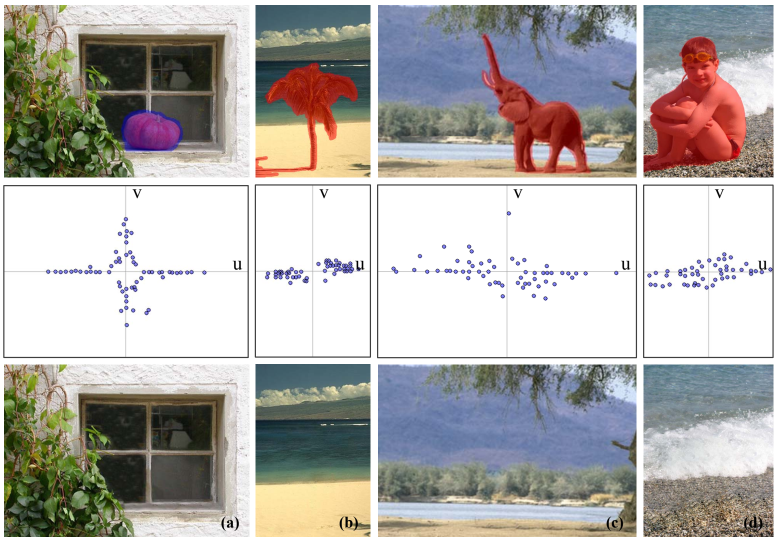

Fig. 4. Linear structures. Top: input. Middle: dominant offsets. Bottom: our results. (a) Long/thin objects. (b) Linear color edges. (c)(d) Linear textural edges.

It is worth mentioning that the non-nearby constraint ($|\mathbf{s}| > \tau$) in (1) is important. A recent work on natural image statistics [20] 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 $(0, 0)$ (Fig. 2(d)). Although the offsets distribution is even sparser (see Fig. 2(a)), the zero offset is insignificant for inferring the structures in the hole.

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.
Linear structures include the cases of long and thin objects (Fig. 4(a)), color edges (Fig. 4(b)), and textural edges (Fig. 4(c)(d)). In these examples, we use a

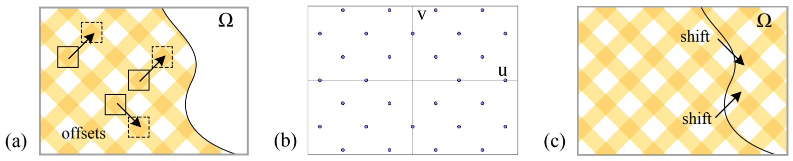

Fig. 5. Illustration of completing textures. The notations are as in Fig. 3.

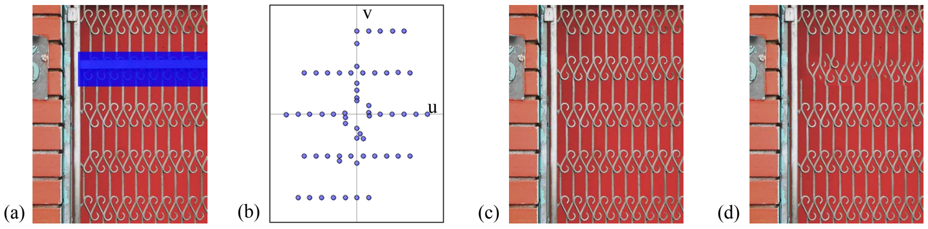

Fig. 6. Regular textures. (a): input. (b): dominant offsets. They describe how the textures are repeated. (c): our result. (d): result of Content-Aware Fill.

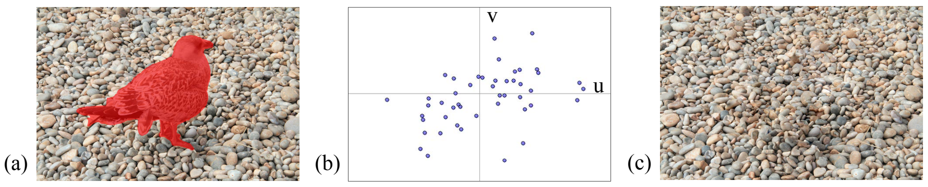

Fig. 7. Random textures. (a): input. (b): dominant offsets. (c): our result.

dot to denote a dominant offset in the histogram. As illustrated in Fig. 3(a), 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. 3(b)), as observed in the real examples in Fig. 4. We can complete this linear structure by shifting the image along this direction (Fig. 3(c)). Note that our method is tolerant to the structures that are not salient (*e.g.*, Fig. 4(c)) or that are not strictly straight (*e.g.*, Fig. 4(d)); it is sufficient if the structures have a “trend” along one or a few directions.

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. 5). This is observed in real images (Fig. 6). We can complete the texture by shifting the image using these offsets. On the other hand, an *irregular* texture generates random patterns (Fig. 7). In this case our method behaves just like the Graphcut texture algorithm [17].

Repeated objects can also generate prominent peaks in the offset statistics. This is helpful in synthesizing semantic content. As shown in Fig. 8, 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

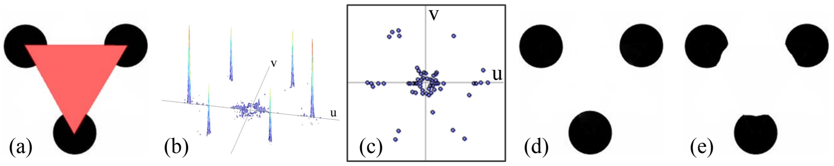

Fig. 8. Repeated circles. (a): input with a missing region in red. (b): offset histogram. (c): dominant offsets found by our algorithm. (d): our result (this is a real result, not a synthetic illustration). (e): result of Content-Aware Fill.

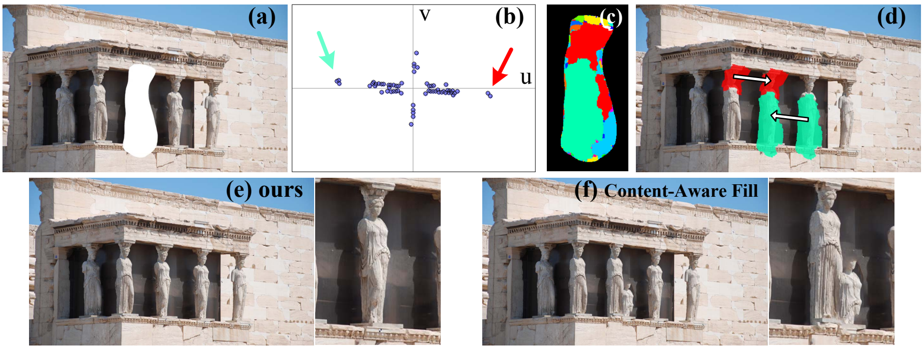

Fig. 9. Repeated objects. (a): input with a sculpture missing. (b): dominant offsets. The arrows indicate the offsets of the relative positions of the sculptures. (c): the label map obtained by graph-cuts: each color represents an offset. (d): the hole is mainly filled by copying the other sculptures, using the offsets indicated in (b). (e) our result and zoom-in. (f) result of Content-Aware Fill and zoom-in.

offsets. In Fig. 9 we show a real example in which our algorithm faithfully recovers a fully missing sculpture. We find that matching-based methods (like [9]) might produce unsatisfactory results (see Fig. 8(e) and Fig. 9(f)), mainly because they are unaware of how the objects are repeated.

Our method has the advantage that it need not consider the above cases separately. It can handle all of them or a mixture of them in the same framework.

3.3 Offsets Sparsity and Optimized Energy

Our method has an energy function (3) similar to the Shift-map method [15]. The main difference is that Shift-map allows all possible offsets. Thus 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 is very often unsatisfactory even if their energy is much lower than ours. In Fig. 10 we optimize the energy (3) respectively using our sparse offsets (Fig. 10(b)) and using all possible offsets (Fig. 10(c)). The later is the way of Shift-map [15] (except that [15] has an extra gradient smoothness term).

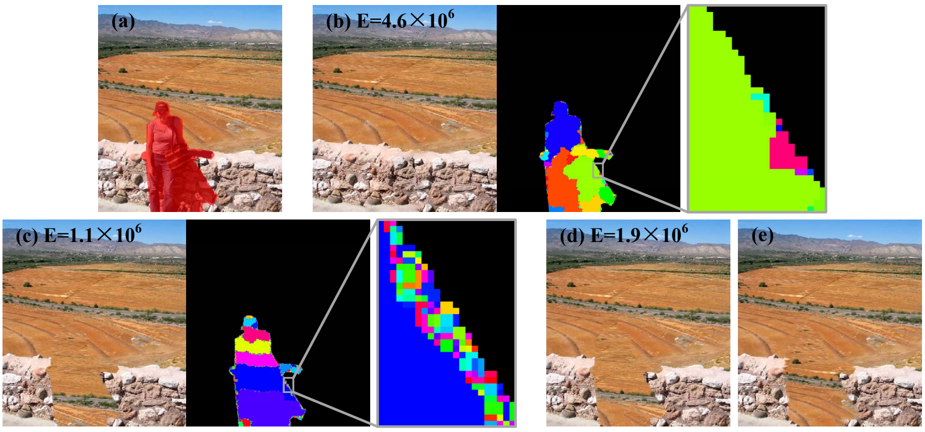

Fig. 10. Offsets sparsity and optimized energy. (a): input. (b): our result and the label map. Energy: E=4.6 $\times$ 10^6, running time: t=0.5s. (c): the result using all possible offsets. E=1.1 $\times$ 10^6, t=4300s. (In this case the number of labels is K=3.8 $\times$ 10^5, and the number of unknown pixels is N=2.2 $\times$ 10^4.) (d): the result of a hierarchical solver [15]. E=1.9 $\times$ 10^6, t=83s. (e): the result from the authors' demo [21] (with gradient smoothness terms).

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 (Fig. 10(b)(c)). We find that with a huge number of offsets, the Shift-map method decreases the energy by "inserting" a great many insignificant labels into the seam (see the zoom-in of Fig. 10(c)). These labels correspond to a few isolated pixels that occasionally "connect" the content on both sides of the seam. But 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 [15] (Fig. 10(d)) or by combining the gradient smoothness term (Fig. 10(e), obtained from the authors' demo [21]). On the contrary, our method is less influenced by this problem (Fig. 10(b)). Actually, exemplar-based methods tend to 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 *limiting the solution space is important for improving the quality in image completion.*

Although a small number of offsets can improve 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. 11. We generate the same number (K=60) of offsets, either on a regular grid (Fig. 11(b)), randomly (Fig. 11(c)), or by our method (Fig. 11(d)). We observe that the alternative methods cannot produce satisfactory results, because the

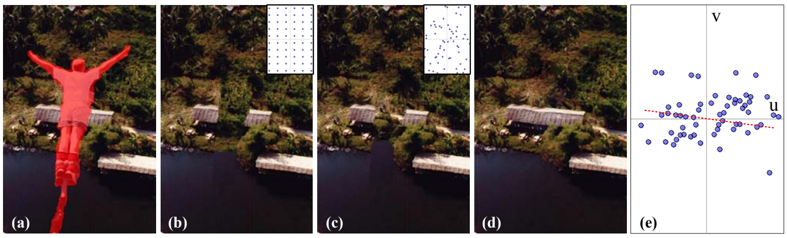

Fig. 11. Comparisons of offsets selection methods. (a): input. (b): the result of regularly spaced offsets. (c): the result of randomly distributed offsets. (d): our result. (e): our offsets. The dash line indicates the offsets used to complete the structure of the roof.

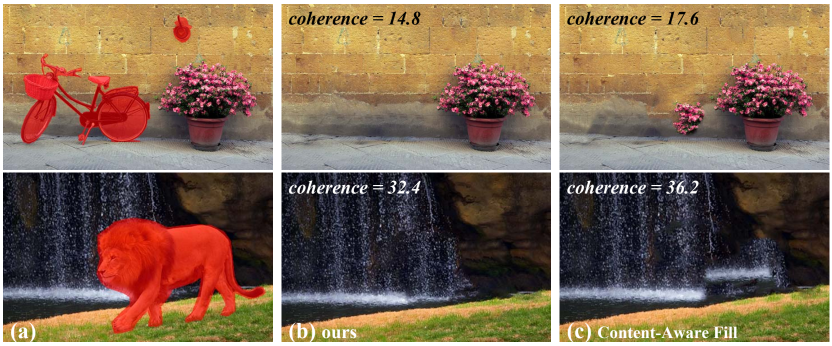

Fig. 12. (a) input. (b) our results. (c) the results of [9], obtained via Content-Aware Fill. The coherence measure is given in each image (computed before Poisson blending).

predefined offsets do not capture sufficient information to predict the missing structures.

3.4 Coherence Measure

Interestingly, we also observe that our method often leads to a better coherence measure [9] even though it does not explicitly optimize this measure. The coherence measure in [9] is defined as:

$$d_{\text{coherence}} = \frac{1}{N} \sum_{P \in \Omega} \min_{Q \in \overline{\Omega}} \|P - Q\|^2, \tag{5}$$

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.

The method in [9] explicitly optimizes the coherence measure by the EM (Expectation-Maximization) algorithm. Because the solution space when defined at the single finest scale is very large and very irregular, the optimization in [9] adopts a multi-scale fashion to avoid bad local minima. We empirically observe that in this manner a few offset values take major effect in the final solution, and they are determined in coarser scales and hardly changed in finer scales. Thus the multi-scale fashion in [9] also plays a role of *limiting the solution space*[2].

On the contrary, our method directly selects the offsets beforehand. In Fig. 12 we show two examples of our method and [9]. We find that our coherence measure is smaller. This indicates that our statistics-based offset selection works better in these cases to reduce the coherence measure. This partially explains why our method has better quality than [9], as in Fig. 12 and many other examples.

4 Implementation Details

Matching similar patches. To efficiently compute $\mathbf{s}$ in (1), we apply a recent nearest-neighbor field algorithm [23] where we additionally reject any patch that disobeys the constraint in the search procedure. This step takes <0.1s for an 800×600 region. We use 8×8 patches and perform matching in a rectangle that is 3 times larger (in length) than the bounding box of the hole. The threshold τ in Eq.(1) is 1/15 of the max of the rectangle's width and height.
Finding dominant offsets. The histogram in (2) is further smoothed by a Gaussian filter ($\sigma=\sqrt{2}$). A "peak" in the smoothed histogram is a bin whose magnitude is locally maximal in a $d\times d$ window ($d=8$).
Combining shifted images. The time complexity of graph-cuts [18] is $O(NK)$, where N is the number of unknown pixels and K is the number of labels. The time of our graph-cuts step is 0.2-0.5 seconds for a 800×600 image with 10-20% pixels missing. As a comparison, it takes over one hour to solve such an image in full resolution if all possible offsets are allowed ($K=10^4$-10^6, like Fig. 10(c)), or tens of seconds using a hierarchical solver [15] with the coarsest level as small as 100×100 pixels (like Fig. 10(d)). Thus our method is 1-2 orders of magnitude faster than Shift-map [15].

Our method is scalable for multi-mega-pixel images because K is fixed. But in practice we find it unnecessary to optimize in full resolution. We downsample the image (by a scale l) to 800×600 and apply the entire algorithm. Then we upsample the resulting label map to full resolution by nearest-neighbor interpolation and multiply the offsets by l. To correct small misalignments we optimize (3) in full resolution, but allow each pixel to take 5 offsets: the upsampled offset and 4 relative offsets[3]. We only solve for the pixels that are in $\frac{l}{2}$-pixel around the seams. This upsampling takes <0.1s for typical 2Mp images using the code in [24]. We have also tested our algorithm in full resolution without downsampling, and found the visual qualities are similar.

A Poisson fusion [25] is applied to further hide small seams. We adopt a recent $O(N)$ time Poisson solver [26]. In our implementation it takes 50ms/Mp.

[2] Contemporary with our work, a recent paper [22] also observes that the method of [9] updates mainly around the boundaries of coherent regions at finer scales. This means that the majority of the used offsets have been limited in the coarser scales.
[3] If the upsampled shift is $\mathbf{s}=(u,v)$, then the other 4 shifts are $(u\pm\frac{l}{2},v)$ and $(u,v\pm\frac{l}{2})$.

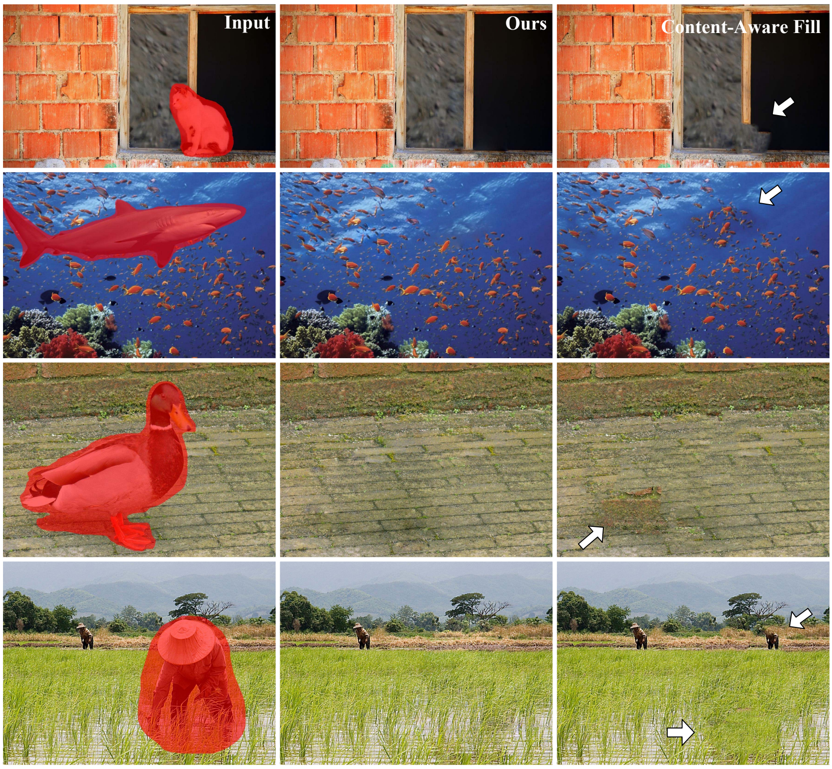

Fig. 13. Comparisons with Content-Aware Fill. Left: input. Middle: our results. Right: results of Content-Aware Fill. The artifacts are highlighted by the arrows. Image size (from top to down): 0.6Mp, 2Mp, 4Mp, 10Mp. The running time is in Fig. 14.

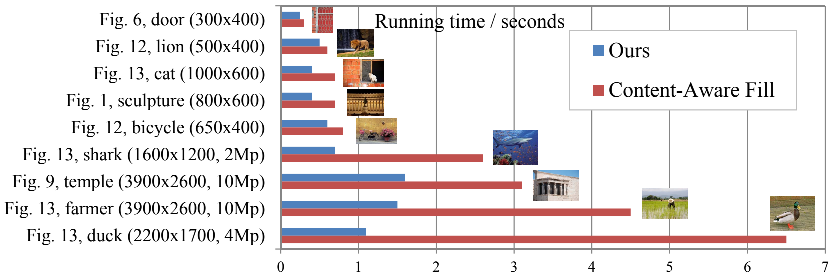

Fig. 14. Running timing comparisons with Content-Aware Fill. The bars are sorted in the ascending order of Content-Aware Fill's time. The two-scale solution does not take effect in the first five examples.

5 Experimental Results

In all the experiments in this paper, we strictly fix all the parameters. This shows the robustness of our algorithm in various cases. Note some other methods

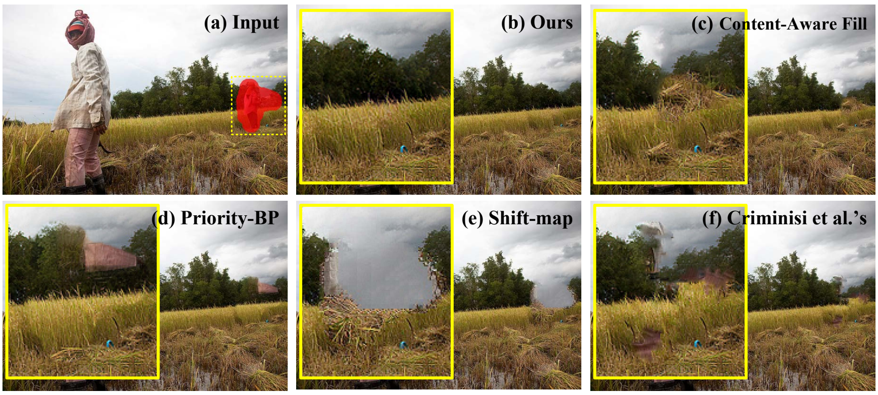

Fig. 15. Comparisons with state-of-the-art methods. (a) Input (640×430). (b) Ours (0.18s). (c) Content-Aware Fill (0.3s). (d) Priority-BP [28] (117s). (e) Shift-map [15] (13s). (f) Criminisi *et al.*'s [7] (6.9s).

require manually tuned parameters for each individual image. Our algorithm is also very robust to various patch sizes (see the supplementary material).

Comparison with Content-Aware Fill. We believe *Content-Aware Fill* in Photoshop [13] is a well-tuned, optimized, and perhaps enhanced version of [9, 12]. Thus we compare with [9, 12] by this implementation[4]. The comparisons are in Fig. 13 and also in Fig. 1, 6, 8, 9, 12 (more in supplementary materials).

In Fig. 14 we compare the speed with Content-Aware Fill on a PC with an Intel Core i7 3.0GHz CPU and 8G RAM. Both methods are using quad cores[5]. For small images where the two-scale solution does not take effect (the first five examples in Fig. 14), our method is slightly faster than Content-Aware Fill. For mega-pixel images our method is 2-5 times faster.

Comparisons with other state-of-the-art methods. In Fig. 15 we further compare with Priority-BP [28], Shift-map [15], 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. 15). More comparisons are in Fig. 16, Fig. 17, and in the supplementary materials.

Limitations. Our method may fail when the desired offsets do not form dominant statistics. Fig. 18(a-c) show our failure case. We can partially solve this problem by manually introducing offsets. *E.g.*, we can paint an extra stroke on the image (Fig. 18(d)), and treat this image as a new source for patch statistics. This stroke contributes to the statistics and overcomes the problem (Fig. 18(e)). Some other failure examples are in the supplementary materials.

[4] We have also tested [9, 12] using the code in [27], but we find it is non-trivial to tune universally acceptable parameters.

[5] Our method benefits *less* than Content-Aware Fill in multi-core, mainly because the EM algorithm and PatchMatch are fully parallelized but the graph-cut is not. In our method, the parallelism is mainly for matching patches and Poisson blending.

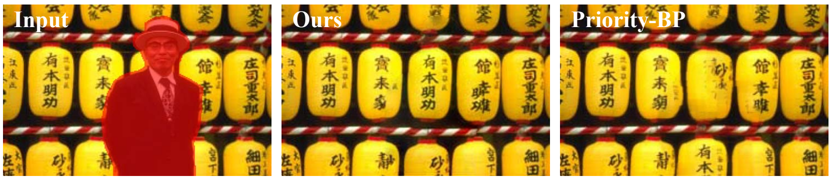

Fig. 16. Comparison with Priority-BP [14]. For this 256×163 image Priority-BP takes 40s in a well parallelized quad-core implementation, while our method takes 0.09s. Also note Priority-BP cannot recover the pattern in the lamp.

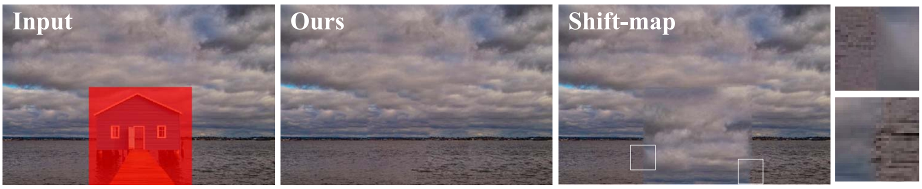

Fig. 17. Comparison with Shift-map [15]. In the zoom-in we show how Shift-map behaves near inconsistent seams. This result is obtained from the authors' demo [21]. We have tried various parameter settings but observed similar artifacts.

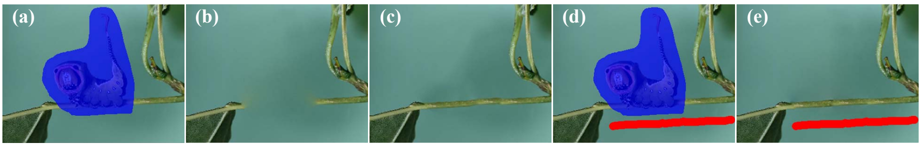

Fig. 18. Failure case. (a) Input. (b) Our result. (c) Result of Content-Aware Fill. (d) An extra stroke is casually painted on the image. (e) Our result of (d).

6 Conclusion

Image statistics have been proven essential in computer vision. Gradient-domain statistics have been applied in denoising [29], deconvolution [30], and inpainting [3], whereas recent patch-domain statistics have been shown very successful in denoising [29] and super-resolution [31]. Our method, which is based on the internal statistics of patch offsets, can be further studied in such context.

References

1. Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C.: Image inpainting. SIGGRAPH (2000) 417–424
2. Ballester, C., Bertalmio, M., Caselles, V., Sapiro, G., Verdera, J.: Filling-in by joint interpolation of vector fields and gray levels. TIP (2001) 1200–1211
3. Levin, A., Zomet, A., Weiss, Y.: Learning how to inpaint from global image statistics. ICCV (2003) 305–312

4. Bertalmio, M., Vese, L., Sapiro, G., Osher, S.: Simultaneous structure and texture image inpainting. CVPR (2003)
5. Roth, S., Black, M.J.: Fields of experts: a framework for learning image priors. CVPR (2005) 860–867
6. Efros, A.A., Leung, T.K.: Texture synthesis by non-parametric sampling. ICCV (1999) 1033–1038
7. Criminisi, A., Perez, P., Toyama, K.: Object removal by exemplar-based inpainting. CVPR (2003)
8. Drori, I., Cohen-Or, D., Yeshurun, H.: Fragment-based image completion. SIGGRAPH (2003) 303–312
9. Wexler, Y., Shechtman, E., Irani, M.: Space-time video completion. CVPR (2004)
10. Sun, J., Yuan, L., Jia, J., Shum, H.Y.: Image completion with structure propagation. SIGGRAPH (2005) 861–868
11. Simakov, D., Caspi, Y., Shechtman, E., Irani, M.: Summarizing visual data using bidirectional similarity. CVPR (2008) 1–8
12. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: PatchMatch: A randomized correspondence algorithm for structural image editing. SIGGRAPH (2009) 1–8
13. Adobe: (`www.adobe.com/technology/graphics/content_aware_fill.html`)
14. Komodakis, N., Tziritas, G.: Image completion using efficient belief propagation via priority scheduling and dynamic pruning. TIP (2007) 2649–2661
15. Pritch, Y., Kav-Venaki, E., Peleg, S.: Shift-map image editing. ICCV (2009)
16. Agarwala, A., Dontcheva, M., Agrawala, M., Drucker, S., Colburn, A., Curless, B., Salesin, D., Cohen, M.: Interactive digital photomontage. SIGGRAPH (2004)
17. Kwatra, V., Schödl, A., Essa, I., Turk, G., Bobick, A.: Graphcut textures: image and video synthesis using graph cuts. ACM SIGGRAPH 2003 Papers (2003)
18. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. TPAMI (2001) 1222–1239
19. Liu, T., Sun, J., Zheng, N.N., Tang, X., Shum, H.Y.: Learning to detect a salient object. CVPR (2007)
20. Zontak, M., Irani, M.: Internal statistics of a single natural image. CVPR (2011)
21. Shift-map On-line Demo: (www.vision.huji.ac.il/shiftmap/)
22. Darabi, S., Shechtman, E., Barnes, C., Goldman, D.B., Sen, P.: Image Melding: Combining Inconsistent Images using Patch-based Synthesis. SIGGRAPH (2012)
23. He, K., Sun, J.: Computing nearest-neighbor fields via propagation-assisted kd-trees. CVPR (2012)
24. Multi-label Graph Cuts: (http://vision.csd.uwo.ca/code/)
25. Pérez, P., Gangnet, M., Blake, A.: Poisson image editing. SIGGRAPH (2003)
26. Farbman, Z., Fattal, R., Lischinski, D.: Convolution pyramids. SIGGRAPH Asia (2011) 175:1–175:8
27. Mansfield, A., Prasad, M., Rother, C., Sharp, T., Kohli, P., Gool, L.V.: Transforming image completion. BMVC (2011)
28. Komodakis, N., Tziritas, G.: Image completion using global optimization. CVPR (2006) 442–452
29. Katkovnik, V., Foi, A., Egiazarian, K., Astola, J.: From local kernel to nonlocal multiple-model image denoising. IJCV (2010) 1–32
30. Fergus, R., Singh, B., Hertzmann, A., Roweis, S.T., Freeman, W.T.: Removing camera shake from a single photograph. SIGGRAPH (2006) 787–794
31. Glasner, D., Bagon, S., Irani, M.: Super-resolution from a single image. ICCV (2009)