Patch Mosaic for Fast Motion Deblurring

Hyeoungho Bae,[1] Charless C. Fowlkes,[2] and Pai H. Chou [1]

[1] EECS Department, University of California, Irvine
[2] Computer Science Department, University of California, Irvine

Abstract. This paper proposes using a mosaic image patches composed of the most informative edges found in the original blurry image for the purpose of estimating a motion blur kernel with minimum computational cost. To select these patches we develop a new image analysis tool to efficiently locate informative patches we call the *informative-edge map*. The combination of *patch mosaic* and informative patch selection enables a new motion blur kernel estimation algorithm to recover blur kernels far more quickly and accurately than existing state-of-the-art methods. We also show that patch mosaic can form a framework for reducing the computation time of other motion deblurring algorithms with minimal modification. Experimental results with various test images show that our algorithm to be 5-100 times faster than previously published blind motion deblurring algorithms while achieving equal or better estimation accuracy.

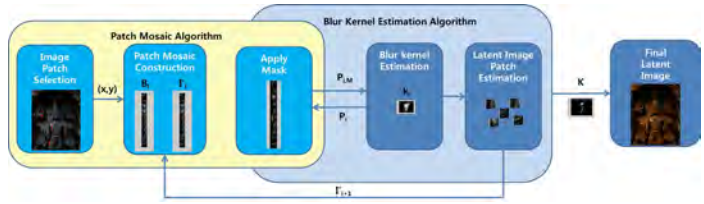
1 Introduction

Motion blur due to relative motion between the camera and the scene during camera exposure plagues consumer photographs, particularly under low-light conditions. Methods that remove such blur from a single photograph are of great practical interest. Thanks to recent advances in image deblurring algorithms, it is now possible to recover unblurred images from blurry sources caused by motion that is not compensated by the optical or mechanical image-stabilization devices integrated in modern digital cameras. However, due to the intensive computational requirements, the motion-blur deconvolution process of an HD-sized image typically takes several to tens of minutes, thereby precluding their adaptation in mainstream consumer products. The goal of this research is to develop a computationally cheap blind-motion deconvolution algorithm with good performance.

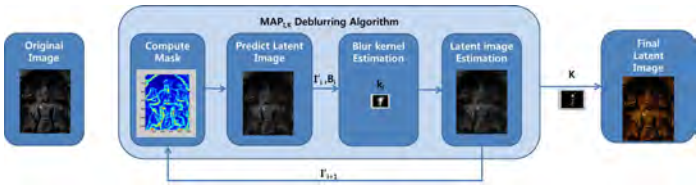
1.1 Fast Blind Motion Deblurring and Patch Mosaics

The motion blur can be modeled as a point spread function (or a *motion blur kernel*) for each point in the image. The relation between the blurry image B, the latent image I, the motion blur kernel k, and noise n can be defined by Equation (1):

$$B(x, y) = I \otimes k_{x,y} + n \tag{1}$$



(a) Patch mosaic-based fast motion-deblurring



(b) Conventional $MAP_{I,k}$ deblurring

Fig. 1. Patch mosaic-based fast motion-deblurring vs Conventional $MAP_{I,k}$ deblurring algorithm: (a) The patch mosaic-based fast motion-deblurring algorithm - Patch mosaic can significantly reduces the computation time of the deblurring. (b) Conventional $MAP_{I,k}$ deblurring algorithm - The patch mosaic algorithm can easily be applied to most of recent $MAP_{I,k}$ deblurring algorithms.

The motion blur can be due to camera motion relative to the scene (e.g. camera shake), objects moving within the scene (e.g. running children in the scene), or both. In such scenarios, the blur kernel is different for each pixel. However, estimating a spatially-variant blur kernel imposes a significant computational burden, especially for high-resolution images. For blur generated from small, translational motions of a camera that is imaging a distant scene, it is often assumed that the kernel is spatially-invariant (i.e. $k_{x,y} = k$). In the remainder of the paper we will focus on this spatially-invariant case.

The typical approach to estimating the latent image is to find a combination (I, k) that minimizes the difference between the two sides of Equation (1). In a probabilistic setting, one can either estimate the kernel k while marginalizing over the latent image I or jointly estimate a kernel and latent image (termed MAP_k and $MAP_{I,k}$ respectively by Levin et al., [1]). MAP_k usually takes significantly more computation than $MAP_{I,k}$ due to marginalization requirements for computing $p(k|B)$ from $p(I, k|B)$. However, as Levin et al. point out, the $MAP_{I,k}$ approach exhibits a strong tendency to return a *'no-blur'* solution in which $I = B$. The $MAP_{I,k}$ approach has still been shown useful in blind image deconvolution [2–7] although evidence of the *'no-blur'* preference can be found in some of the works using a standard image sparsity prior in their energy function (e.g., [2]). One technique we will exploit for overcoming the no-blur bias of simple image priors is to use the gradient of the reference image (e.g., the original blurry image) as a spatial prior for the gradient of the estimated latent image [3, 4].

Our approach to blur-kernel estimation makes two contributions. The first is the novel *patch mosaic*, which summarizes the informative edges found in a

blurry image. We construct the patch mosaic by tiling informative image patches to synthesize a new, compact blurry image. Using the patch mosaic, we can effectively reduce the blur-kernel estimation time. Since only the informative part of the image is used, there is no need to calculate sophisticated masks for occluding unwanted parts (saturated or filled with narrow edges) of the image for every single iteration such as those found in [4, 6].

To locate patches containing informative edges efficiently, we develop a tool named *informative-edge map*. It can locate edges satisfying several attributes of informative edges including contrast, orientation angle, straightness [8], and usefulness [4]. Through the performance evaluation, we show that the informative-edge map locates the most appropriate area within the blurry image to estimate the blur kernel and yields the most accurate results while consuming only 4% of the computation time of the kernel estimation process.

We integrated these two components into a new, fast motion-deblurring algorithm. The computation time of our algorithm is 5 to 40 times faster than those in the comparison group, while the benchmark estimation accuracy is the best among the group, which is shown in Section 3. Our algorithm implemented in MATLAB takes 15 seconds to estimate a 2256×1504-pixel latent color image on a Intel Core i7 processor. Since our framework (shown in Fig. 1) is compatible with most of the recent $MAP_{I,k}$ motion-deblurring algorithms, we can apply the patch mosaic framework as a generic tool for reducing computation time of other blind motion-deblurring algorithms. To demonstrate the feasibility of this idea, we modified the deblurring algorithm by Krishnan et al. [9]. The results show that the modified algorithm runs 4.7 times faster than the original algorithm with similar estimation accuracy.

1.2 Related Work

To our best knowledge, there has been no previous work that uses an image patch mosaic for blur-kernel estimation. We briefly discuss two closely related lines of work: patch-based image analysis and fast motion deblurring.

Patch-based image analysis: Jia suggested using small fractions of an image for blind deconvolution [5]. However, their work needs user intervention for selecting appropriate areas and foreground and background colors for alpha-blending calculation. A larger number of patches increases the complexity of the energy function for estimating the blur kernel and latent image. If there is little distinction between the foreground and background objects, the algorithm yields inaccurate estimation results as shown in [3]. Joshi et al. analyzed the edge profile of an image to predict the latent edge [6]. This is related to the filter-based sharp edge prediction method as in [2–4], but is limited to blurry edges caused by simple motions orthogonal to the edge. In contrast, our measure seeks patches that individually contain many different orientation angles. In the area of spatially variant motion-deblurring algorithms, Gupta et al. estimated the camera trajectory by patch-based analysis of the scene [10]. However, they applied the blind deconvolution algorithm of [3] for the entire image area first. Then, they used RANSAC-based approach to select appropriate results among

initially deblurred image patches to reconstruct the camera motion. We use our image-patch locating algorithm to find the most informative image patches before estimating the blur kernel. Our method requires neither user intervention during the selecting process nor computationally expensive initial deconvolution step.

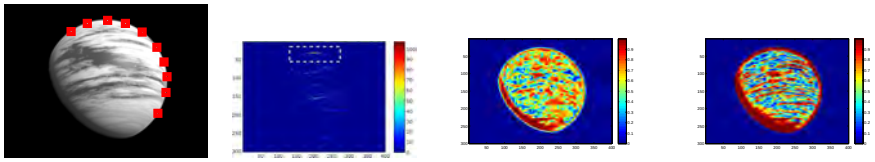
Fast motion deblurring algorithms: Several works address reducing the computation time of blind motion-deblurring algorithms [2,4,6,11]. Cho and Lee take the $MAP_{I,k}$ approach and use discrete Fourier transform to reduce the computation time. However, they still need to use the whole image area to estimate the blur kernel, which requires their compiled C++ binary 4 to 6 times more computation time compared to our interpreted Matlab script. As mentioned earlier, their algorithm also seems to prefers *'no-blur'* results for some cases. Even though the accuracy of Xu and Jia's algorithm is remarkable, the computation time of their C++ binary code is more than 27 times slower than our Matlab script due to the sophisticated masking and the larger number of iterations to increase the accuracy [4]. By using the patch mosaic, we can achieve even better accuracy, as quantitative analysis will show in Section 3. Krishnan et al. take a $MAP_{I,k}$ approach but with l_1/l_2 regularization to predict the latent image in a coarse-to-fine iteration [11]. However, they still need more than 5 minutes to estimate a color latent image of 558×858 pixels. The authors of [4] has developed a fast deblurring software using GPU. However, our work is targeting more light-weight platforms like smart phones or portable cameras. So we don't think that kind of approach can be regarded as our competitor since it requires huge computation resources compared to ours. There is also work on speeding up non-blind deconvolution [9, 12] where the kernel is known. Such work could be used as a final step in combination with our approach for quickly estimating the kernel.

2 Blur-Kernel Estimation Algorithm

The blur-kernel estimation algorithm (Fig. 1) is composed of four parts: (1) *Image-patch selection*, (2) *Patch-mosaic construction*, (3) *Blur-kernel estimation* and (4) *Latent image-patch estimation*. In image-patch selection, the algorithm finds a set of patches that cover all possible edge-orientation angles and are likely to be informative for estimating blur. After selecting the image patches, the selected image patches are combined to construct the patch mosaic for the blur-kernel estimation step. We estimate the kernel from this mosaic using a coarse-to-fine iteration where in each step the estimated blur kernel is deconvolved with the individual image patches to estimate the latent image patches. These deblurred patches are then upsampled and used as the image prior for the next iteration.

2.1 Image-Patch Selection

The main concern of our approach is how to reduce the amount of data to process without sacrificing estimation accuracy. As described in several publications,



(a) Located patches (b) *IE map* (c) Straightness map (d) Usability map

Fig. 2. Selected image area: (a) The location of selected image patches . (b) The informative edge map from Equation (2). The edges of highest *IE* value located in the rectangle box with angle between $-9°$ and $9°$. (c) The straightness map (d) The usability map.

edges found in the original blurry image are particularly useful for estimating the blur kernel caused by motion. There are several important characteristics of edges in motion deblurring; gradient magnitude, orientation angle, width of the edges, and straightness of the edges [2–8, 13]. Applying masks on top of the blurry image is a common approach to make the estimation algorithm focus on the most informative areas in the image [4, 6, 8, 10]. However, this does not reduce the actual amount of data to be processed. Moreover, computation of complex masks takes significant additional time [4, 8]. For this reason, some previous works require user intervention to select the most informative area in the image [9, 13]. By using the patch mosaic, our approach avoids the need to construct a sophisticated mask for the whole image.

Fig. 2 shows an illustrative example of the image-patch selection process. We use the gradient magnitude of a downsampled image to locate the sharpest edges. The blurred trajectory of an edge provides information for estimating one-dimensional motion that is perpendicular to the edge orientation. To secure sufficient information about all possible directions of motion and to recover the full 2D kernel, it is necessary to analyze edges that span the full range of orientations. We start with a pixel-wise measure of informativeness that incorporates our desired criteria. We specify the informativeness of a pixel by

$$IE = M \circ (sR > \tau_s) \circ (U > \tau_u), \tag{2}$$

where *IE* means the validity of information of the edge, M is the gradient magnitude of the image, and sR and U are the straightness map from by Cho [8] and usability map by Xu et al. [4], respectively. τ_s and τ_u are the threshold values for filtering non-straight edges and less-usable edges, respectively. To ensure that we can recover motion in any direction, we also ensure diversity in the angles of the edges in selected patches. We define a pixel-wise angle mask that identifies those pixels whose gradient falls in given range of angles

$$angle\_mask_i = Q_n^i \left[\arctan \left(\frac{\bigtriangledown B_y}{\bigtriangledown B_x} \right) \right], \tag{3}$$

where Q_n^i means the quantization function to make the i^{th} angle mask among n groups. For example, in Fig. 2(b), *IE* is masked to pick out angles between $-9°$ and $9°$.

Algorithm 1 Patch Selection

Input: M, sR, U, $angle\_mask$, max_{patch}
$P \leftarrow \{\}$
$A \leftarrow$ set of angles to cover
while $(|P| < max_{patch})\&(|A| > 0)$ **do**
  $IE \leftarrow M \circ \left(\sum_{i \in A} angle\_mask_i\right) \circ (sR > \tau_s) \circ (U > \tau_u)$
  $x \leftarrow$ largest element of IE
  $P \leftarrow P \cup \{x\}$
  $A \leftarrow A - angle(x)$
  $M \leftarrow M - near(x)$
end while
Output: patch locations P

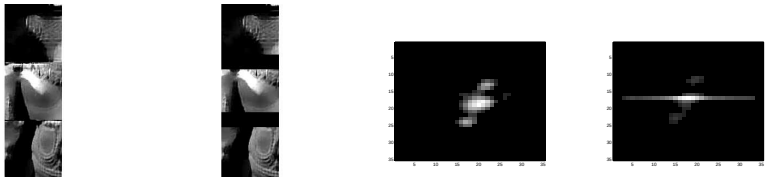
$angle(x)$ is the range of angles in which x falls
$near(x)$ is the elements near the coordinate of x

The pseudo code for selecting image patches is shown in Algorithm 1. During the iteration, the IE map is masked using the orientation angles not yet covered, so that orientation angles already contained in the set of selected image patches will not be included in the sum of the angle mask in Equation (2). Since the usefulness map and the straightness map used in the equation have a high computational cost, we downsample the original image. Unlike [4], our algorithm does not need to use mask for noisy edges or saturated area since we use relatively small areas filled with strong edges of homogeneous orientation angles compared to other algorithms using the entire image area. In our performance evaluation, we choose the size of image patch to be 5 times that of the initial kernel size, which is 3×3 pixels.

2.2 Patch Mosaic Construction

The tricky part of using multiple image patches is estimating a single blur kernel satisfying all the patches. The authors of [5] introduced multiple alpha-blending variables into the energy function, while [8] merge using the inverse Radon transform. However, these approaches are computationally expensive, and we have found that the latter introduces noise into the final result. Instead, we merge the patches to make a single patch mosaic that contains sufficient information to estimate the blur kernel with minimum computation cost (see Fig. 3(a)). Since we use a single image, we do not need to introduce multiple variables into the energy function. Even though our patch mosaic is composed of patches containing single orientation angles, we do not suffer from noise caused by the 1-D blur integration profile analysis and inverse Radon transform for merging the kernel estimation of individual patches. Instead, the merging happens implicitly in the de-convolution by the constraint of the kernel being spatially invariant.

Simply tiling the image patches from different areas of an image introduces a discontinuity of gradient between adjacent patches and introduces errors in the recovered kernel (shown in Fig. 3(a)). To solve this problem, we mask out



(a) Patch mosaic (b) With mask (c) With mask (d) Without mask

Fig. 3. The mask for image patch boundary: (a) Patch mosaic without the mask. (b) Patch mosaic with the mask. (c) Estimated blur kernel using the boundary mask. (d) Estimated blur kernel without using the mask.

Algorithm 2 Coarse-to-fine blur kernel estimation using the patch mosaic I

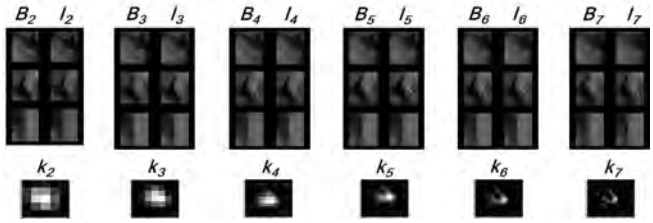
INPUT: O,x,y
Let O_i be **O** downsampled to **scale(i)**
Sample O_1 to get image patches $p_1^1, ...p_1^m$ using **x,y**
Construct the blurry image patch mosaic B_1 using $p_1^{j\in 1,\ldots,m}$
Filter the patches and construct the initial latent patch mosaic I_1
for $i = 1 \ldots nscales$ **do**
  Estimate blur kernel $\hat{k}$ from I_i and B_i by minimizing Eqn. (4)
  Estimate latent patches $\hat{l}_i^{j\in 1,\ldots,m}$ from B_i, $\hat{k}$, I_i by minimizing Eqn. (6)
  Upsample and filter $\hat{l}_i^{j\in 1,\ldots,m}$ to get I_{i+1}
  Construct the blurry image patch mosaic B_{i+1}
end for
OUTPUT: $\hat{k}$

O is the original blurry image
x,y are the image patch coordinates from the image patch selector

the boundaries between the patches (Fig. 3(b)). This is done by assigning zero weight to the appropriate terms in the error function used for estimating the kernel (see Equation (4)). Stacking the patches vertically minimizes the masked region. Since the most important information of each patch lies in the middle of the patch, we can mask without losing the estimation accuracy (Fig. 3(c)).

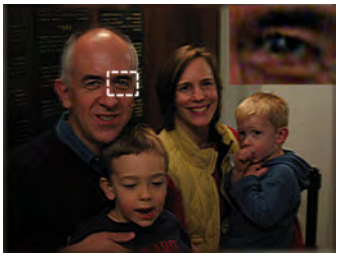
2.3 Blur-Kernel Estimation

Blur-kernel estimation is carried out in a coarse-to-fine manner, starting with a downsampled version of the original blurry image. The main advantage of this approach is that it can effectively keep the estimation algorithm from falling into local minima [2–4, 13]. The original image is downsampled to several different resolutions. The algorithm starts to estimate the blur kernel on the downsampled version of the image patches. It estimates the latent image patches using the resulting low-resolution blur-kernel estimate. These estimated latent image patches are then upsampled and provide the latent image prior used in the next iteration. Since edges in the upsampled images are necessarily missing high frequencies, we apply sharpening using bilateral and shock filtering to localize the



(a) Patch mosaic during the kernel estimation

(b) Original image



(c) Estimated latent image

Fig. 4. Patch mosaic based blur kernel estimation: (a) Patch mosaic during the kernel estimation - B_i, I_i, and k_i represent the blurry patch mosaic, latent patch mosaic and the estimated blur kernel, respectively. (b) The original blurry image (c) The estimated latent image

edges. As mentioned previously, image-patch selection is also carried out on the low-resolution version of the blurry image.

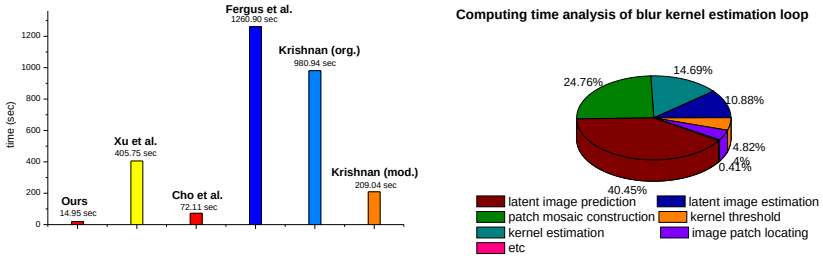
Given an estimate of the latent image I, the energy function used for estimating the blur kernel is given by:

$$\hat{k} = \arg\min_{k} \left[\|\bigtriangledown I \otimes k - \bigtriangledown B\|_M^2 + \gamma \|k\|^2 \right], \tag{4}$$

where I is the latent estimation for the patch mosaic, k is the blur kernel, and B is the original patch mosaic. The squared error is only computed on those unmasked pixels specified by the mask M. We set the coefficient (γ) value as 0.001. During the blur kernel estimation process, we used energy function having L2 norm form, which is computationally cheap. According to Plancherel's theorem, the Fourier transform is an isometry so we can analyze the squared error in the frequency domain. This means that minimizing Equation (4) is an identical process to minimizing the frequency domain version of it [14].

$$\hat{k} = \mathcal{F}^{-1} \left(\frac{\overline{\mathcal{F}(\bigtriangledown \bar{I}_{x,M})} \circ \mathcal{F}(\bigtriangledown B_{x,M}) + \overline{\mathcal{F}(\bigtriangledown \bar{I}_{y,M})} \circ \mathcal{F}(\bigtriangledown B_{y,M})}{\overline{\mathcal{F}(\bigtriangledown \bar{I}_{x,M})} \circ \mathcal{F}(\bigtriangledown \bar{I}_{x,M}) + \overline{\mathcal{F}(\bigtriangledown \bar{I}_{y,M})} \circ \mathcal{F}(\bigtriangledown \bar{I}_{y,M}) + \gamma} \right), \tag{5}$$

where $\bar{I}$ means the predicted latent image from the blurry image. To prevent the boundary artifact, the mask mentioned in Section 2.2 is applied after taking the gradient. We use the predicted latent image from the previous iteration as



(a) Computation time (b) Blur kernel estimation loop

Fig. 5. Computation time analysis: (a) Computation time comparison between the algorithms. (b) The computation time analysis during the blur-kernel estimation loop.

a prior for the latent-image estimation to avoid the bias towards the *'no-blur'* solution (discussed in Section 1).

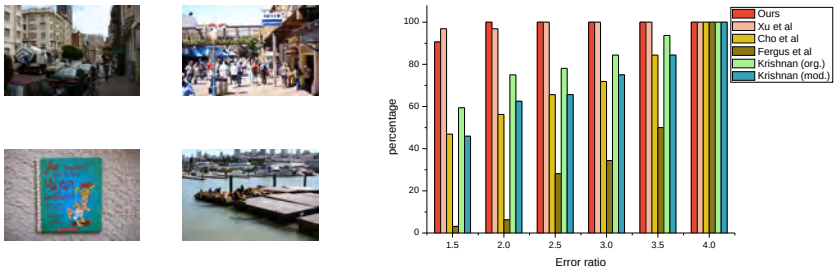
$$\hat{I} = \arg\min_{I} \left[\|I \otimes \hat{k} - B\|^2 + \omega \|\bigtriangledown I - \bigtriangledown \bar{I}\|^2 \right] \tag{6}$$

Since Equation (6) is used to estimate individual image patches (the latent image patch estimation part of Fig. 1(a)), the boundary mask is not required for the equation. Since we process only small portions of the entire image, the time required to get $\hat{k}$ and $\hat{I}$ is far less than the entire image area. Detailed timing analysis will be provided in Section 3.1. After estimating the blur kernel, we use the equation (6) to get the latent image. For the color image, we use the same blur kernel for each color channel, (R, G, B). Algorithm 2 gives the pseudo code of the blur-kernel estimation process. For the evaluation in Section 3.2, we used 7 iterations of the loop in running Algorithm 2. The size of the images is upscaled to $\sqrt{2}$ of that of the previous stage. The example of the patch mosaic based blur-kernel estimation is shown in Fig. 4. As illustrated in Fig. 1(a), the patch mosaic of blurry image (B_i) and the patch mosaic of latent image (I_i) are made for each iteration to estimate the blur kernel (k_i).

To minimize the equation (4) and (6), we use fast fourier transform approach, which can introduce the ringing artifact around high frequency components in the blurry image (especially when minimizing the equation (6)). The two main reasons of the artifact are: 1) the discontinuity at the boundary of the image, and 2) inaccurate blur kernel (also mentioned in [3]). To address the first reason, we taper the edge of the boundary of the blurry image patch using Gaussian filter. After estimating the latent image for the next stage, we applied the boundary mask on the left and right side of the patch to prevent the artifact from spreading into the center of the patch (as shown in Fig. 4).

3 Results

We compare performance of our proposed algorithm with several competitive blind spatially invariant motion-blur estimation algorithms including the works



(a) Test images (b) Error ratio comparison

Fig. 6. Performance comparison: (a) Test images: From the top-left corner in clockwise order, "Downtown", "Fisherman", "Sealion", and "Storybook", (b) Error ratio comparison.

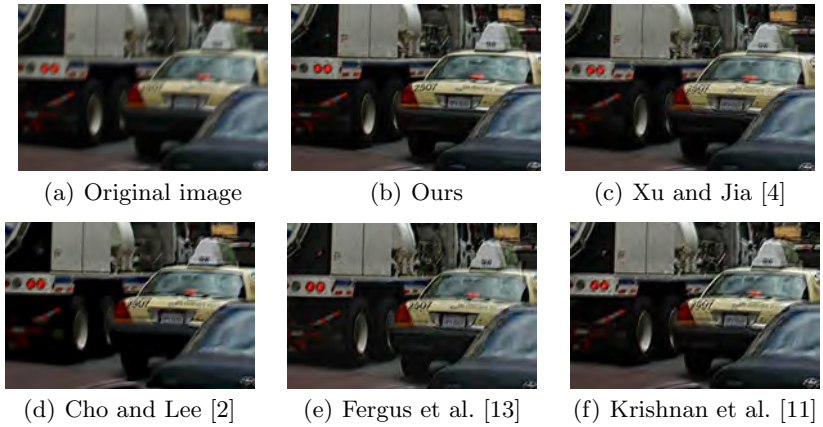
of Fergus et al., Krishnan et al., Cho and Lee, and Xu and Jia [2, 4, 11, 13]. The evaluation is based on three criteria: (1) computation time, (2) quantitative analysis on the estimation accuracy using the test images with known blur kernels, and (3) tests for real-world images [1].

3.1 Computation Time

We compared the computation time with the competitive algorithms using the test images of 2256×1504 resolution used in Section 3.2 (Fig. 5). Our and Fergus's algorithms are written in Matlab script. Xu and Jia's and Cho and Lee's algorithms are C/C++ compiled binaries. The experimental platform was a laptop with an Intel Core-i7 processor with 6 GB RAM. In Fig. 5(a), the average computation time of our algorithm is 14.95 seconds, which is 4.8 times faster than the compiled binary version of the fastest algorithm (Cho and Lee's). Our algorithm is 27 times faster than Xu and Jia's and 84 times faster than Fergus et al.'s. Since our algorithm is not a C/C++ compiled binaries, we expect a compiled version of our algorithm will achieve even faster runtime.

Fig. 5(b) shows the timing breakdown of our algorithm: 44% (6.55 sec) is used for the final latent-image estimation for calculating with Equation (6) from [15] and 53% of the time is used for our blur-kernel estimation algorithm. The most time-consuming part is the latent-image prediction, which uses bilateral/shock filters for sharpening the upsampled latent image patches in each iteration. The second largest portion is used for the patch-mosaic construction step. Only 25.5% of the time is used for calculating Equations (4) and (6) during the estimation process. It is also worth noting that once the kernel has been estimated, the non-blind deblurring of the whole image takes a significant proportion of the time (43.4%, 6.5 sec from 14.95 sec). The patch location and mosaic construction take only 14.8% of the total time for the whole deblurring process. If we consider

[1] The executable of our algorithm can be downloaded from `http://cecs.uci.edu/~hyeoungho/image_deblur.html`



(a) Original image (b) Ours (c) Xu and Jia [4]

(d) Cho and Lee [2] (e) Fergus et al. [13] (f) Krishnan et al. [11]

Fig. 7. Image deblurring results of one blurry "Downtown" image: (a) is the cropped area from the original blurry image.

the time reduction in the blur-kernel estimation process enabled by the patch mosaic, the portion will be much less than that. We thus achieve more than 400% performance improvement in computation time over competitive algorithms for kernel estimation.

3.2 Estimation Accuracy

For the quantitative analysis of the accuracy of estimated blur kernel, we use the analysis methodology of Levin et al. [16]. We use the kernels that Levin et al. used in their work [2] and the four images of 2256×1504 pixels shown in Fig. 6(a) to make 32 different blurry images. Fig. 6(b) compares the performance of the four algorithms including ours.[3] *SSDE* stands for the sum of squared differences between the two images. The error ratio is used to evaluate the accuracy of estimated blur kernels independently of the error introduced by deconvolution. For this purpose, Richardson-Lucy non-blind image deconvolution algorithm is used for the latent image estimation [17, 18].

Generally, our algorithm yields the most accurate blur kernels compared to other competitive algorithms in the group. The blur kernels estimated by our algorithm have the error ratio of less than 2 over all the test images. For 62.5% of the test images (20 out of 32), the estimation accuracy of our algorithm is better than Xu and Jia's. Considering the fact that our script is 27 times faster than their compiled binary code, this is a remarkable improvement. In Fig. 7, one of the deblurring results from those algorithms are provided. The images are the results of their own latent image estimation algorithms. All the images are close-ups from the whole images. The original blurry image is shown in Fig. 7(a). The estimated latent images using the suggested algorithm, Xu and Jia's, Cho and Lee's and Fergus et al.'s are provided in Figs. 7(b) through 7(e), respectively.

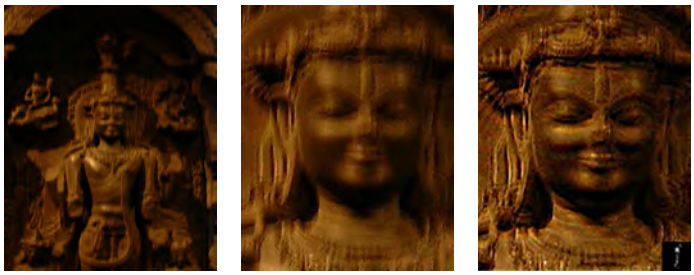
[2] `www.wisdom.weizmann.ac.il/~levina/papers/LevinEtalCVPR09Data.zip`

[3] We fixed the parameters of all the algorithms during the evaluation.

(a) Original (b) Zoomed-in (c) Ours

(d) Xu and Jia (e) Cho and Lee (f) Krishnan et al.

Fig. 8. *'pietro.jpg'* deblur results (from [11])



(a) Original image (b) Zoomed-in (c) Ours

(d) Xu and Jia (e) Cho and Lee (f) Fergus et al.

Fig. 9. 'Lyndsey2.jpg' from [13]

3.3 Real-World Images

In this section, we show some deblurred results from the real-world blurry images from different blind deconvolution papers. The estimated latent-image results are from their own non-blind deconvolution algorithms. In Fig. 8, we compared our result with Xu and Jia, Cho and Lee, and Krishnan et al. [11]. Ours and Krishnan et al.'s estimated the most accurate blur kernels among the comparison groups. The light that is reflected in the eye (top-right corner of Fig. 8(c) - Fig. 8(f)) is effectively deblurred to be a spot in both Figs. 8(c) and 8(f). The estimated blur kernels from Xu and Jia's and Cho and Lee's algorithms are not accurate

compared to our result; the deblurred light in the eye is not a spot (Xu and Jia's result shown in Fig. 8(d)) or unobservable (Cho and Lee's result in Fig. 8(e)).

In Fig. 9, we compare our result with those of Xu and Jia, Cho and Lee, and Fergus et al. using the image from [13]. The estimated blur kernels are shown at the right-bottom side of each image. In this case, Cho and Lee's algorithm shows a preference to *'no-blur'* result in their kernel estimation. Moreover, the latent image shows some loss of edges. On the other hand, the result of Xu and Jia shows larger noise compared to other results. Considering their estimated kernel is similar to ours, the artifact can be attributed to their non-blind deconvolution algorithm.

3.4 Patch Mosaic as a General Framework

To show that our patch mosaic can be used to reduce the computation time of other motion deblurring algorithms, we selected Krishnan et al's algorithm [9][4]. We used the approach illustrated in Fig. 1(a). The image that is fed into the estimation algorithm is synthesized by the patch mosaic algorithm for every step of the coarse-to-fine iteration of their algorithm. The performance of the modified algorithm is shown in Fig. 5 and 6. On average, the modified algorithm is 4.7 times faster than the original algorithm with the cost of 17.8% increased error ratio. The average error ratio of the modified algorithm is 2.13, which is slightly higher than the original algorithm of 1.81 (see Krishnan (org.) vs. Krishnan (mod.) in Figs. 5(a) and 6(b)). However, the computation time of the modified one is only 209 sec compared to 980 sec of the original algorithm.

4 Conclusions

We propose the *patch mosaic* and a fast, accurate deblurring algorithm based on the patch mosaic for spatially invariant blur-kernel estimation. A MATLAB implementation of our deblurring algorithm runs 4.7 times faster than the nearest competitor (the C++ binary of [2]) when estimating a latent image of 2256×1504 pixels. Our algorithm achieves equal or better accuracy than other algorithms published to date. The patch mosaic can not only improve the computation time but also exclude saturated and uniform image regions that typically mislead the estimation process [4, 13]. We also show that the patch mosaic can be used for other deblurring algorithms to reduce the computation time with minimal loss of the estimation accuracy. The time analysis shows that our patch mosaic-based algorithm can effectively reduce the computation time with minimal overhead. Since more than 40% of the time in the estimation loop is used for the latent image prediction, faster filtering solutions may yield further increases in computation speed [19]. Our algorithm may also be combined with external sensors to further reduce the computation time and provide stronger priors [20]. For example, a triaxial accelerometer can be used to aid the image

[4] `http://cs.nyu.edu/~dilip/research/blind-deconvolution/`

patch locating algorithm to focus on the edges sensitive to the direction of the motion detected by the sensor. Our algorithmic ideas can also be extended to solving for more complex spatially variant blur-kernel estimation problems in a timely manner [10].

Acknowledgement. This work was sponsored by the National Science Foundation grant CBET-0933694 and Air Force Office of Scientific Research grant FA9550-10-1-0538. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. Levin, A., Weiss, Y., Durand, F., Freeman, W.T.: Efficient marginal likelihood optimization in blind deconvolution. CVPR (2011)
2. Cho, S., Lee, S.: Fast motion deblurring. ACM Transactions on Graphics **28** (2009)
3. Shan, Q., Jia, J., Agarwala, A.: High-quality motion deblurring from a single image. ACM Transactions on Graphics **27** (2008)
4. Xu, L., Jia, J.: Two-phase kernel estimation for robust motion deblurring. ECCV (2010) 157–170
5. Jia, J.: Single image motion deblurring using transparency. CVPR (2007)
6. Joshi, N., Szeliski, R., Kriegman, D.: Psf estimation using sharp edge prediction. CVPR (2008)
7. Yuan, L., Sun, J., Quan, L., Shum, H.: Image deblurring with blurred/noisy image pairs. ACM Transactions on Graphics **26** (2007)
8. Cho, T.: Motion blur removal from photographs. M.I.T Ph.D dissertation (2010)
9. Krishnan, D., Fergus, R.: Fast image deconvolution using hyper-laplacian priors. Neural Information Processing Systems (2009)
10. Gupta, A., Joshi, N., Zitnick, L., Cohen, M., Curless, B.: Single image deblurring using motion density functions. In: ECCV '10: Proceedings of the 10th European Conference on Computer Vision. (2010)
11. Krishnan, D., Tay, T., Fergus, R.: Blind deconvolution using a normalized sparsity measure. CVPR (2011)
12. Wang, Y., Yang, J., Yin, W., Zhang, Y.: A new alternating minimization algorithm for total variation image reconstruction. SIAM Journal of Imaging Science **1** (2009)
13. Fergus, R., Singh, B., Hertzmann, A., Roweis, S.T., Freeman, W.T.: Removing camera shake from a single photograph. ACM Transactions on Graphics **25** (2006)
14. Bracewell, R.N.: The Fourier Transform and Its Applications. McGraw-Hill (1999)
15. Levin, A., Fergus, R., Durand, F., Freeman, W.T.: Deconvolution using natural image priors. http://groups.csail.mit.edu/graphics/CodedAperture /SparseDeconv-LevinEtAl07.pdf (2007)
16. Levin, A., Weiss, Y., Durand, F., Freeman, W.T.: Understanding and evaluating blind deconvolution algorithms. CVPR (2009)
17. Richardson, W.H.: Bayesian-based iterative method of image restoration. Journal of Optics (1972)
18. Lucy, L.B.: An iterative technique for the rectification of observed distributions. Astronomical Journal (1974)
19. Yang, Q., Tan, K., Ahuja, N.: Real-time $o(1)$ bilateral filtering. CVPR (2009)
20. Joshi, N., Kang, S.B., Zitnick, C.L., Szeliski, R.: Image deblurring using inertial measurement sensors. ACM Transactions on Graphics **29** (2010)