Fast Motion Deblurring

Sunghyun Cho
POSTECH

Seungyong Lee
POSTECH

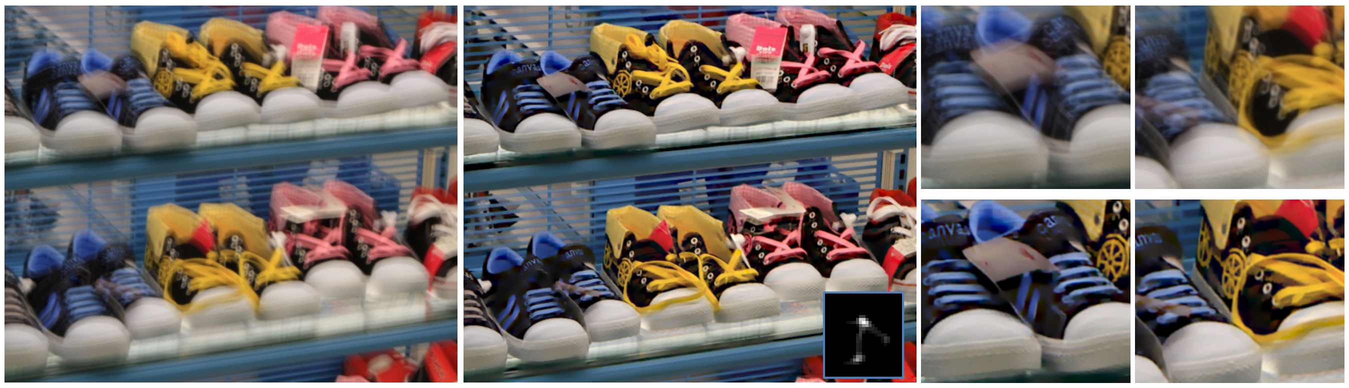

Figure 1: *Fast single image deblurring. Our method produces a deblurring result from a single image very quickly. Image size:* 713×549*. Motion blur kernel size:* 27×27*. Processing time:* 1.078 *seconds.*

Abstract

This paper presents a fast deblurring method that produces a deblurring result from a single image of moderate size in a few seconds. We accelerate both latent image estimation and kernel estimation in an iterative deblurring process by introducing a novel prediction step and working with image derivatives rather than pixel values. In the prediction step, we use simple image processing techniques to predict strong edges from an estimated latent image, which will be solely used for kernel estimation. With this approach, a computationally efficient Gaussian prior becomes sufficient for deconvolution to estimate the latent image, as small deconvolution artifacts can be suppressed in the prediction. For kernel estimation, we formulate the optimization function using image derivatives, and accelerate the numerical process by reducing the number of Fourier transforms needed for a conjugate gradient method. We also show that the formulation results in a smaller condition number of the numerical system than the use of pixel values, which gives faster convergence. Experimental results demonstrate that our method runs an order of magnitude faster than previous work, while the deblurring quality is comparable. GPU implementation facilitates further speed-up, making our method fast enough for practical use.

CR Categories: I.4.3 [Image Processing and Computer Vision]: Enhancement—Sharpening and deblurring

Keywords: motion blur, deblurring, image restoration

1 Introduction

A motion blur is a common artifact that produces disappointing blurry images with inevitable information loss. It is caused by the nature of imaging sensors that accumulate incoming lights for an amount of time to produce an image. During exposure, if the camera sensor moves, a motion blurred image will be obtained.

If a motion blur is shift-invariant, it can be modeled as the convolution of a latent image with a motion blur kernel, where the kernel describes the trace of a sensor. Then, removing a motion blur from an image becomes a deconvolution operation. In *non-blind* deconvolution, the motion blur kernel is given and the problem is to recover the latent image from a blurry version using the kernel. In *blind* deconvolution, the kernel is unknown and the recovery of the latent image becomes more challenging. In this paper, we solve the blind deconvolution problem of a single image, where both blur kernel and latent image are estimated from an input blurred image.

Single-image blind deconvolution is an ill-posed problem because the number of unknowns exceeds the number of observed data. Early approaches imposed constraints on motion blur kernels and used parameterized forms for the kernels [Chen et al. 1996; Chan and Wong 1998; Yitzhaky et al. 1998; Rav-Acha and Peleg 2005]. Recently, several methods were proposed to handle a more general motion blur given a single image [Fergus et al. 2006; Jia 2007; Shan et al. 2008]. While these methods can produce excellent deblurring results, they necessitate intensive computation. It usually takes more than several minutes for the methods to deblur a single image of moderate size.

Most blind deconvolution methods take an iterative process that alternatingly optimizes the motion blur kernel and the latent image. In the process, the blur kernel is obtained from the estimated latent image and the given blurred image. The kernel is then used to estimate the latent image by applying non-blind deconvolution to the given blurred image. The new estimated latent image is used for kernel estimation in the next iteration. The intensive computation of previous methods stems from the complicated methods used for kernel estimation and latent image estimation. Optimization involving large matrices and vectors is needed for kernel estimation, and sophisticated optimization techniques are necessary to handle non-blind deconvolution with non-linear priors.

This paper presents a fast blind deconvolution method that produces a deblurring result from a single image in only a few seconds. The high speed of our method is enabled by accelerating both kernel estimation and latent image estimation steps in the iterative deblurring process. Our method produces motion deblurring results with comparable quality to previous work, but runs an order of magnitude faster. A C++ implementation of our method usually takes less than one minute to deblur an image of moderate size, which is about 20 times faster than the C-language implementation of the previous method [Shan et al. 2008]. A GPU implementation of our method further reduces the processing time to within a few seconds, which is fast enough for practical applications of deblurring.

To accelerate latent image estimation, we introduce a novel prediction step into the iterative deblurring process. Strong edges are predicted from the estimated latent image in the prediction step and then solely used for kernel estimation. This approach allows us to avoid using computationally inefficient priors for non-blind deconvolution to estimate the latent image. Small deconvolution artifacts could be discarded in the prediction step of the next iteration without hindering kernel estimation. For non-blind deconvolution, we use a simple method with a Gaussian prior, which can be computed quickly using Fourier transforms. Since only simple image processing techniques are used in the prediction step, the combination of simple non-blind deconvolution and prediction can efficiently obtain an estimated latent image used for kernel estimation.

For kernel estimation, we formulate the optimization function using image derivatives rather than pixel values. We take a conjugate gradient (CG) method to solve the numerical system derived from the optimization function, and use Fourier transforms to calculate the gradient needed for the CG method. Working with image derivatives allows us to reduce the number of Fourier transforms from twelve to two, saving $5/6$ of the computation required for calculating the gradient. In addition, we show that the condition number of the numerical system using image derivatives is smaller than that using pixel values, which enables the CG method to converge faster. Consequently, the numerical optimization process for kernel estimation is significantly accelerated in our method.

The contributions of this paper can be summarized as follows.

- We propose a fast method for single-image blind deconvolution, which deblurs an image of moderate size within a few seconds.
- We accelerate latent image estimation in the iterative deblurring process, without degrading the accuracy of kernel estimation, by combining a prediction step with simple non-blind deconvolution.
- We achieve significant acceleration of the numerical optimization for kernel estimation with formulation using image derivatives rather than pixel values.
- The acceleration technique for kernel estimation can be adopted to other deblurring methods for performance improvement.

2 Related Work

To reduce the ill-posedness of blind deconvolution, many previous approaches made restrictive assumptions on a motion blur, such as a parameterized linear motion in a single or multiple images. Chen *et al.* [1996] used two consecutively blurred images to estimate a motion blur. Rav-Acha and Peleg [2005] used horizontally and vertically blurred images to help reconstruct details. Cho *et al.* [2007] used multiple images with space-variant blurs for finding the blur kernels. For a single image input, Yitzhaky *et al.* [1998] made an isotropy assumption to estimate a motion blur. Dai and Wu [2008] used the blur information obtained by alpha matting to estimate per-pixel blur kernels. Ji and Liu [2008] proposed a method to handle more general types of blur, such as 1D accelerated motions. Due to the assumption of simple parametric blur kernels, these approaches cannot handle the high diversity of a 2D motion blur.

Recently, several algorithms were proposed to handle a more general motion blur. Fergus *et al.* [2006] used a variational Bayesian method with natural image statistics to deblur an image. However, with the complexity of their statistical model, it takes a relatively long time to estimate a PSF, even on a small image patch. Jia [2007] used an alpha matte that describes transparency changes caused by a motion blur for kernel estimation. The method largely depends on the quality of the input alpha matte. Yuan *et al.* [2007] used two images for motion deblurring, of which one is noisy but has sharp edges, and the other is motion blurred. Shan *et al.* [2008] introduced an effective deblurring method using a series of optimization techniques. It however still needs several minutes to deblur an image. Levin *et al.* [2009] analyzed previous blind deconvolution methods and quantitatively evaluated [Fergus et al. 2006] and [Shan et al. 2008] using a data set.

Hardware approaches have also been introduced for image deblurring. Ben-Ezra and Nayar [2004] proposed a hybrid imaging system, where a high-resolution camera captures the blurred frame and a low-resolution camera with a faster shutter speed is used to estimate the camera motion. Tai *et al.* [2008] also built a similar hybrid system to handle a spatially varying motion blur. Levin *et al.* [2008] introduced a system for capturing a uniformly blurred image by controlling the camera motion even if the objects have different 1D movements.

Assuming the blur kernel is known, several non-blind deconvolution methods were proposed, aiming at correctly estimating the latent sharp image [Lucy 1974; Wiener 1964]. Recently, Yuan *et al.* [2008] introduced an effective non-blind deconvolution method, which utilizes bilateral filtering and residual deconvolution to reduce ringing artifacts.

There have been image deblurring methods that include image filtering similar to our prediction step. Money and Kang [2008] used a shock filter to find sharp edges, and estimated a motion blur kernel using the filtered image. As a simple parametric blur kernel is assumed, this method cannot handle a more complicated blur. Joshi *et al.* [2008] predicted sharp edges using edge profiles and estimated motion blur kernels from the predicted edges. However, their goal is to remove small blurs, which can be described with single peaks. Independently of our work, the PhD dissertation of Joshi [2008] mentioned as future work a blind deconvolution method that uses prediction in an iterative process. However, the dissertation does not contain details of the method, and the preliminary results show a far lower quality of deblurring than ours.

3 Fast Single Image Blind Deconvolution

3.1 Single Image Blind Deconvolution

A motion blur is generally modeled as

$$B = K * L + N, \tag{1}$$

where B is a blurred image, K is a motion blur kernel or a point spread function (PSF), L is a latent image, N is unknown noise introduced during image acquisition, and $*$ is the convolution operator. In blind deconvolution, we estimate both K and L from B, which is a severely ill-posed problem.

A successful approach for blind deconvolution is alternating optimization of L and K in an iterative process. In the latent image estimation and kernel estimation steps of the process, we respectively solve the equations similar to

$$L' = \text{argmin}_L\{\|B - K * L\| + \rho_L(L)\}, \tag{2}$$

$$K' = \text{argmin}_K\{\|B - K * L\| + \rho_K(K)\}. \tag{3}$$

In Eqs. (2) and (3), $\|B - K * L\|$ is the data fitting term, for which the L_2 norm is usually used, and ρ_L and ρ_K are regularization terms. For ρ_L, Chan and Wong [1998] used total variation, and Jia [2007] used a term that prefers a two-tone image for the transparency map. Shan *et al.* [2008] used image derivatives as well as pixel values for the data fitting term while a natural image prior is used for ρ_L. Several regularization terms have been used for ρ_K, such as L_1 and L_2 norms, total variation, and a uniform prior that means no regularization.

The main purpose of the iterative alternating optimization is to progressively refine the motion blur kernel K. The final deblurring result is obtained by the last non-blind deconvolution operation that is performed with the final kernel K and the given blurred image B. The intermediate latent images estimated during the iterations have no direct influence on the deblurring result. They only affect the result indirectly by contributing to the refinement of kernel K.

The success of previous iterative methods comes from two important properties of their latent image estimation, *sharp edge restoration* and *noise suppression in smooth regions*, which enable accurate kernel estimation. Although we assume the blur is uniform over the given image, a more accurate blur kernel can be obtained around sharp edges. For example, we cannot estimate a blur kernel on a region with a constant intensity. Since a natural image usually contains strong edges, a blur kernel can be effectively estimated from the edges reconstructed in latent image estimation. Noise suppression in smooth regions is also important because such regions usually occupy much larger areas than strong edges in a natural image. If noise has not been suppressed in smooth regions, the data fitting term in Eq. (3) would be significantly affected by the noise, compromising the accuracy of kernel estimation from strong edges.

To achieve sharp edge restoration and noise suppression when solving Eq. (2), previous methods usually perform computationally expensive non-linear optimization. In addition, kernel estimation using Eq. (3) is computationally demanding as it involves a number of operations on huge matrices and vectors. As a result, both latent image estimation and kernel estimation require heavy computation in the iterative process of previous blind deconvolution methods.

3.2 Fast Blind Deconvolution

In this paper, we present a fast blind deconvolution technique by reducing the computational overhead for latent image estimation and kernel estimation. For accelerating latent image estimation, we assume that the latent image has enough strong edges, and explicitly pursue sharp edge restoration and noise suppression using image filters, instead of taking a computationally expensive non-linear prior in Eq. (2). For kernel estimation, we accelerate the numerical optimization process of Eq. (3) by excluding pixel values in the formulation.

In our method, latent image estimation is divided into two parts: simple deconvolution and prediction. Given a blurred image B and kernel K, we first remove the blur to obtain an estimate L of the latent image using simple and fast deconvolution with a Gaussian prior. Due to the characteristics of a Gaussian prior, L would contain smooth edges and noise in smooth regions. In the prediction step, we obtain a refined estimate L' by restoring sharp edges and removing noise from L with efficient image filtering techniques. As a result, L' provides a high-quality latent image estimation needed for accurate kernel estimation, in spite of the poor quality of simple deconvolution (see Fig. 2).

For kernel estimation, we use a CG method to solve Eq. (3). During the solution process, we need to calculate the gradient of the energy function many times. The gradient calculation requires heavy computation, involving multiplications of huge matrices and vectors. Fortunately, the multiplications correspond to convolution operations and can be accelerated using fast Fourier transforms (FFTs). However, when we perform FFTs in sequence, we should properly handle image boundaries, which prohibits direct concatenation of FFTs. By formulating the energy function in Eq. (3) with only image derivatives, we can simplify the boundary handling and reduce the number of FFTs significantly. The formulation also makes the numerical system derived from Eq. (3) well-conditioned, providing a fast convergence.

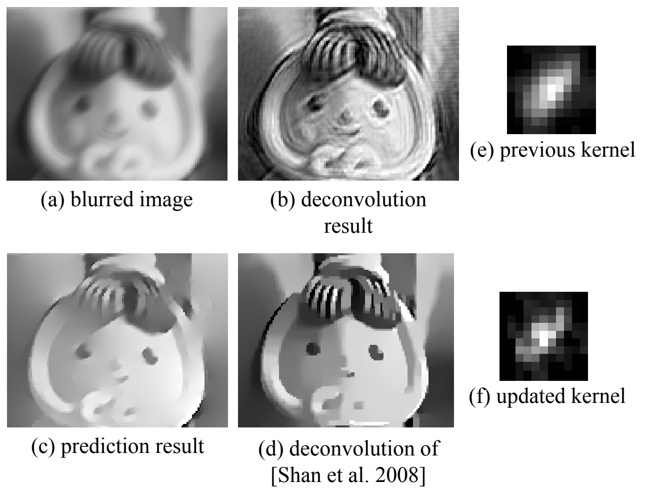

Figure 2: *Effect of prediction. Our simple deconvolution with (a) and (e) produces a poor result (b). However, the prediction result (c) from (b) has only sharp edges while ringing artifacts have been removed. The kernel (f) estimated using (c) shows a refinement toward the real kernel, shown in Fig. 4. Comparison of (c) with (d) obtained directly from (a) and (e) shows that combination of simple deconvolution with prediction can produce a similar result to sophisticated deconvolution, for the purpose of the input for kernel estimation. For visualization, the image in (c) has been restored from the predicted gradient maps by Poisson reconstruction.*

3.3 Process Overview

Fig. 3 shows the overall process of our blind deconvolution method. An algorithm summarizing the process can also be found in the supplementary material. To progressively refine the motion blur kernel K and the latent image L, our method iterates three steps: prediction, kernel estimation, and deconvolution. Recall that our latent image estimation is divided into prediction and deconvolution. We place prediction at the beginning of the loop to provide an initial value of L for kernel estimation, where the input of the prediction is the given blurred image B.

In the prediction step, we compute gradient maps $\{P_x, P_y\}$ of L along the x and y directions which predict salient edges in L with noise suppression in smooth regions. Except at the beginning of the iteration, the input of the prediction step is the estimate of L obtained in the deconvolution step of the previous iteration. In the kernel estimation step, we estimate K using the predicted gradient maps $\{P_x, P_y\}$ and the gradient maps of B. In the deconvolution step, we obtain an estimate of L using K and B, which will be processed by the prediction step of the next iteration.

To make the estimations of K and L more effective and efficient, our method employs a coarse-to-fine scheme. At the coarsest level, we use the down-sampled version of B to initialize the process with the prediction step. After the final estimate of L has been obtained at a coarse level, it is up-sampled by bilinear interpolation and then used for the input of the first prediction step at the next finer level. In our experiments, we performed seven iterations of the three steps at each scale. As detailed in Sec. 4, this coarse-to-fine scheme enables handling of large blurs for which prediction with image filtering may not suffice to capture sharp edges.

In the coarse-to-fine iterative process for updating K and L, we use the gray-scale versions of B and L. After the final K has been obtained at the finest level, i.e., with the input image size, we perform the final deconvolution with K on each color channel of B to obtain the deblurring result. Although any non-blind deconvolution technique can be used for this step, we use the technique proposed in [Shan et al. 2008], which efficiently obtains high quality deconvolution results. Fig. 4 shows an example of our deblurring process with intermediate estimates of K and L.

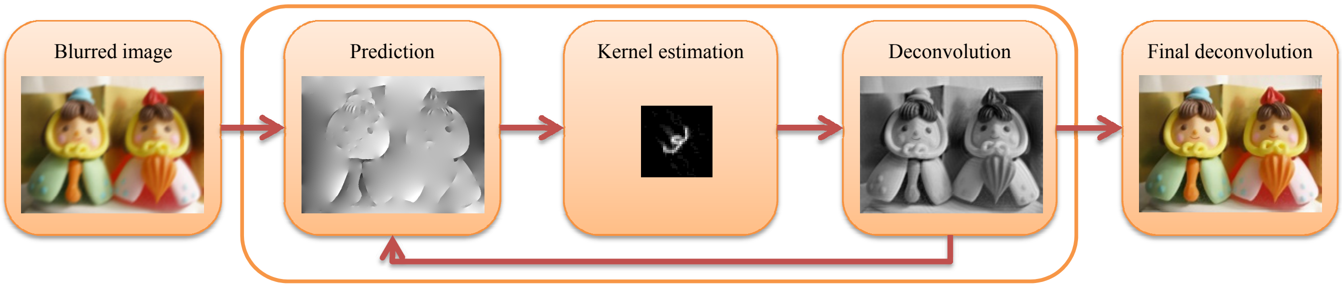

Figure 3: *Overview of our deblurring process.*

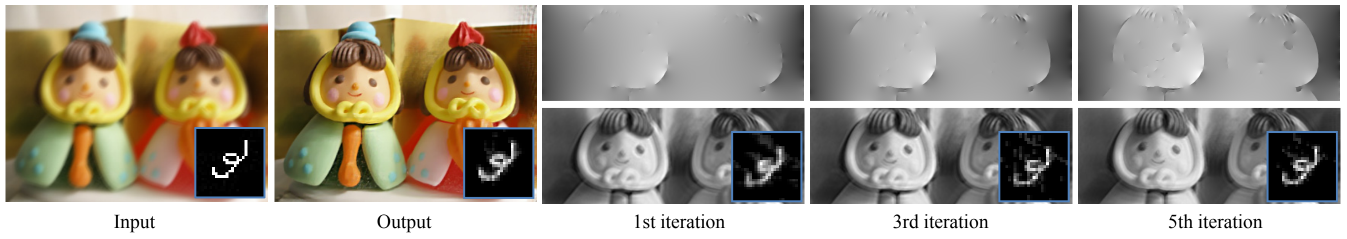

Figure 4: *Deblurring process example. The leftmost image shows the input blurred image and the original motion blur kernel. The second image shows the deblurring result and the final estimated kernel. The remaining three columns show the predicted gradient maps $\{P_x, P_y\}$ (top) and the deconvolution results with estimated kernels (bottom) after different numbers of iterations. For visualizing $\{P_x, P_y\}$, we used Poisson reconstruction. Note that the deblurring process itself does not use Poisson reconstruction of predicted gradients. All intermediate results are shown at the finest scale.*

4 Fast Latent Image Estimation

Prediction In the prediction step, we estimate the image gradient maps $\{P_x, P_y\}$ of the latent image L in which only the salient edges remain and other regions have zero gradients. Consequently, in the kernel estimation step, only the salient edges have influences on optimization of the kernel because convolution of zero gradients is always zero regardless of the kernel.

We use a shock filter to restore strong edges in L. A shock filter is an effective tool for enhancing image features, which can recover sharp edges from blurred step signals [Osher and Rudin 1990]. The evolution equation of a shock filter is formulated as

$$I_{t+1} \quad = \quad I_t - \text{sign}(\Delta I_t)\|\nabla I_t\|dt, \tag{4}$$

where I_t is an image at time t, and ΔI_t and ∇I_t are the Laplacian and gradient of I_t, respectively. dt is the time step for a single evolution.

Our prediction step consists of bilateral filtering, shock filtering, and gradient magnitude thresholding. We first apply bilateral filtering [Tomasi and Manduchi 1998] to the current estimate of L to suppress possible noise and small details. A shock filter is then used to restore strong edges of L. The result L' of shock filtering contains not only high-contrast edges but also enhanced noise. We remove the noise by computing and thresholding the gradient maps $\{\partial_x L', \partial_y L'\}$ of L'. The truncated gradient maps $\{P_x, P_y\}$ give the final output of the prediction step.

The support size of bilateral filtering is fixed as 5×5, and the spatial sigma σ_s is set to 2.0. The range sigma σ_r is a user parameter, which is related to the noise level of the input blurred image B. When B contains much noise, we use a large value for σ_r. For shock filtering, we perform a single evolution of Eq. (4) with the time step dt. At the beginning of the iterative deblurring process, we use large values for σ_r and dt to clearly restore strong sharp edges in L. As the iteration goes, we gradually decrease the values by multiplying 0.9 at each iteration. For most of our experiments, we used 0.5 and 1.0 for the initial values of σ_r and dt, respectively.

The threshold for truncating gradients is determined as follows. To estimate an $m \times m$ kernel, we need the information of blurred edges in at least m different directions. We construct the histograms of gradient magnitudes and directions for each $\partial L'$. Angles are quantized by $45°$, and gradients of opposite directions are counted together. Then, we find a threshold that keeps at least rm pixels from the largest magnitude for each quantized angle. We use 2 for r by default. To include more gradient values in $\{P_x, P_y\}$ as the deblurring iteration progresses, we gradually decrease the threshold determined at the beginning by multiplying 0.9 at each iteration.

Deconvolution In the deconvolution step, we estimate the latent image L from a given kernel K and the input blurred image B. We use the energy function

$$f_L(L) \quad = \quad \sum_{\partial_*} \omega_* \|K * \partial_* L - \partial_* B\|^2 + \alpha\|\nabla L\|^2, \tag{5}$$

where $\partial_* \in \{\partial_o, \partial_x, \partial_y, \partial_{xx}, \partial_{xy}, \partial_{yy}\}$ denotes the partial derivative operator in different directions and orders, $\omega_* \in \{\omega_0, \omega_1, \omega_2\}$ is a weight for each partial derivative, and α is a weight for the regularization term. The first term in the energy is based on the blur model of [Shan et al. 2008], which uses image derivatives for reducing ringing artifacts. The regularization term $\|\nabla L\|^2$ prefers L with smooth gradients, as mentioned in [Levin et al. 2007]. Eq. (5) can be optimized very fast by pixel-wise division in the frequency domain, which needs only two FFTs. For ω_*, we used the values given in [Shan et al. 2008]. We used 0.1 for α.

Optimizing Eq. (5) may not produce high-quality results, compared to sophisticated deconvolution methods (e.g., [Levin et al. 2007; Yuan et al. 2008; Shan et al. 2008]), and the results can contain smoothed edges and ringing artifacts. However, due to the prediction step that sharpens edges and discards small details, this simple deconvolution does not hinder accurate estimation of a blur kernel in our iterative process.

Large blurs Our prediction method may fail to correctly predict sharp edges for large blurs. However, our coarse-to-fine scheme enables us to avoid direct prediction of edges from a largely blurred image. We first predict sharp edges in a low resolution image, where the extents of blurs have been narrowed and most edges can be predicted without severe localization errors. At a higher resolution, we start prediction of sharp edges with an upsampled version of the *deconvolved* image obtained at a coarser resolution, which contains reduced amounts of blurs. In the iterative process at a specific scale, sharp edge prediction is applied to the deconvolution

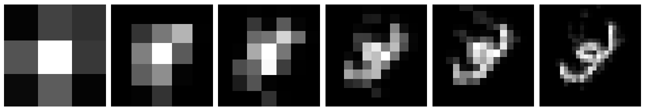

Figure 5: *Estimated kernels at different scales for the deblurring example in Fig. 4. As the scale becomes finer, a more detailed structure of the kernel is recovered.*

result obtained by an updated kernel in the previous iteration, progressively improving the prediction accuracy. With the multi-scale iterative process, we can estimate kernels for large blurs using prediction with small size bilateral and shock filters. This multi-scale iterative process is also the main difference from [Joshi et al. 2008] and [Money and Kang 2008], which enables our method to estimate complex large motion blurs that cannot be handled by the previous methods. Fig. 5 shows estimated kernels at different scales in our multi-scale process.

Speed comparison For estimating latent image gradient maps to be used for kernel estimation, our method requires two FFTs for deconvolution and simple image filtering operations for prediction. Bilateral filtering with a small support size and shock filtering, as well as gradient thresholding, can be performed very fast. In contrast, the highly efficient deconvolution method of Shan *et al.* [2008], which is based on variable substitution, commonly needs 30 to 60 FFTs, i.e., 15 to 30 times more FFTs than our method. Obviously, our method with simple convolution and prediction is much faster than the latent image estimation with a sophisticated deconvolution technique.

5 Fast Kernel Estimation

To estimate a motion blur kernel using the predicted gradient maps $\{P_x, P_y\}$, we minimize the energy function

$$f_K(K) = \sum_{(P_*,B_*)} \omega_* \|K * P_* - B_*\|^2 + \beta\|K\|^2, \quad (6)$$

where $\omega_* \in \{\omega_1, \omega_2\}$ denotes a weight for each partial derivative. P_* and B_* vary among

$$\begin{aligned}(P_*, B_*) \in \{&(P_x, \partial_x B), (P_y, \partial_y B),\\ &(\partial_x P_x, \partial_{xx} B), (\partial_y P_y, \partial_{yy} B),\\ &((\partial_x P_y + \partial_y P_x)/2, \partial_{xy} B)\}. \end{aligned} \quad (7)$$

Each $(K * P_* - B_*)$ forms a map and we define $\|I\|^2 = \sum_{(x,y)}$ I(x,y)^2 for map I, where (x, y) indexes a pixel in I. β is a weight for the Tikhonov regularization.

Our energy function in Eq. (6) is similar to [Shan et al. 2008]. A difference is that we use only the image derivatives, not including the pixel values, in the energy function. In addition, similar to Yuan *et al.* [2007], our energy function includes a Tikhonov regularization term, instead of the L_1 norm of K used in [Shan et al. 2008]. In our experiments, we used the values given in [Shan et al. 2008] for ω_* and set β to 5.

We can write Eq. (6) in a matrix form,

$$\begin{aligned} f_k(\mathbf{k}) &= \|\mathbf{Ak} - \mathbf{b}\|^2 + \beta\|\mathbf{k}\|^2 \\ &= (\mathbf{Ak} - \mathbf{b})^T(\mathbf{Ak} - \mathbf{b}) + \beta\mathbf{k}^T\mathbf{k}, \end{aligned} \quad (8)$$

where $\mathbf{A}$ is a matrix consisting of five P_*'s, $\mathbf{k}$ is a vector representing the motion blur kernel K, and $\mathbf{b}$ is a vector consisting of five B_*'s. To minimize Eq. (8), we use a CG method. Then, the gradient of f_k, defined by

$$\frac{\partial f_k(\mathbf{k})}{\partial \mathbf{k}} = 2\mathbf{A}^T\mathbf{Ak} + 2\beta\mathbf{k} - 2\mathbf{A}^T\mathbf{b}, \quad (9)$$

should be evaluated many times in the minimization process.

Computation of $\partial f_k(\mathbf{k})/\partial\mathbf{k}$ is time consuming due to the vast size of $\mathbf{A}$. When the sizes of L and K are $n \times n$ and $m \times m$, respectively, the size of $\mathbf{A}$ is $5n^2 \times m^2$. Thus, direct computation of $\mathbf{Ak}$ needs heavy computational and storage overhead. Although the size of $\mathbf{A}^T\mathbf{A}$ is $m^2 \times m^2$, which is relatively small, precomputing $\mathbf{A}^T\mathbf{A}$ is still time consuming. Each element of $\mathbf{A}^T\mathbf{A}$ requires the computation of dot product for two shifted versions of five P_*'s.

However, since $\mathbf{Ak}$ corresponds to convolution between five P_*'s and K, we can accelerate the computation by FFTs. Specifically, computing $\mathbf{Ak}$ needs six FFTs: one $\mathcal{F}(K)$ and five $\mathcal{F}^{-1}[\omega_*\mathcal{F}(P_*)\circ\mathcal{F}(K)]$'s, where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the forward and inverse Fourier transforms, respectively, and $\circ$ is a pixel-wise multiplication. Note that we can precompute $\mathcal{F}(P_*)$ before starting the CG method. Similarly, computing $\mathbf{A}^T\mathbf{y}$ can be accelerated by performing six FFTs, where $\mathbf{y} = \mathbf{Ak}$. As a result, it costs a total of 12 FFTs to compute the gradient $\partial f_k(\mathbf{k})/\partial\mathbf{k}$ at each iteration of the CG method. In addition to $\mathcal{F}(P_*)$, $\mathbf{A}^T\mathbf{b}$ can be computed with FFTs in the preprocessing step.

To further accelerate the computation by reducing the number of FFTs, we concatenate evaluations of $\mathbf{Ak}$ and $\mathbf{A}^T\mathbf{y}$ to directly compute $\mathbf{A}^T\mathbf{Ak}$. Then, $\mathbf{A}^T\mathbf{Ak}$ can be computed by

$$\mathcal{F}^{-1}\left[\sum_{P_*} \omega_* \overline{\mathcal{F}(P_*)} \circ \mathcal{F}(P_*) \circ \mathcal{F}(K)\right] \quad (10)$$

with appropriate cropping and flipping operations, where $\overline{\mathcal{F}(P_*)}$ is the complex conjugate of $\mathcal{F}(P_*)$. In Eq. (10), $\sum_{P_*}\omega_*\overline{\mathcal{F}(P_*)}\circ\mathcal{F}(P_*)$ can be precomputed before CG iteration. Thus only two FFTs are needed for computing the gradient, saving 10 FFTs.

This efficient computation benefits from using only image derivatives but no pixel values in Eq. (6). If we include pixel values in Eq. (6), the computation of $\mathbf{Ak}$ with FFTs will have boundary artifacts due to the periodicity property of Fourier transforms. Then, we should handle the boundary artifacts before computing $\mathbf{A}^T\mathbf{y}$. This intervening boundary handling prohibits direct computation of $\mathbf{A}^T\mathbf{Ak}$ using Eq. (10). In contrast, since our method uses only image derivatives, we can avoid the boundary artifacts by simply padding the boundaries of the derivative images P_*'s with zeros before computing $\mathbf{Ak}$. We set the width of a padded image as a power of prime numbers, 2, 3, 5, and 7, which is greater than or equal to $(n + m - 1)$, where n and m are the widths of the input image and the kernel, respectively. The height of a padded image is determined similarly. FFTs can be computed quickly for such image sizes.

Motion blur kernels are generally assumed normalized and containing no negative components. After optimizing Eq. (6), we set elements with values smaller than $1/20$ of the biggest one to zero. Then, the remaining non-zero values are normalized so that their sum becomes one.

Convergence speed In numerical optimization, the number of iterations for convergence, or convergence speed, is important. Interestingly, our method for kernel estimation shows faster convergence than the method including pixel values. In Fig. 6, we used the CG method to estimate the kernel for a synthetically blurred image. The graphs in Fig. 6 show that the kernel estimation error of our method drastically decreases only in a few iterations while the error decreases slowly in the case of using pixel values.

The fast convergence of our method results from the well-conditioned structure of the matrix $\mathbf{A}^T\mathbf{A}$ in Eq. (9). $\mathbf{A}^T\mathbf{A}$ can be represented by

$$\mathbf{A}^T\mathbf{A} = \sum_* \omega_*\mathbf{A}_*^T\mathbf{A}_*, \quad (11)$$

where $\mathbf{A}_*^T\mathbf{A}_*$ is defined by $(\mathbf{A}_*^T\mathbf{A}_*)_{(i,j)} = (\mathbf{l}_*^i)^T(\mathbf{l}_*^j)$. $\mathbf{l}_*^i$ is the vector representation of $\partial_* L$ after shifted by the amount depending

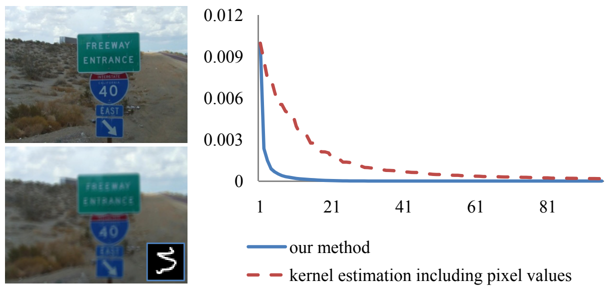

Figure 6: *Convergence speed of the numerical method in kernel estimation. Left: original image and a blurred version used for experiment. Right: error of the intermediate kernel estimate (vertical) vs. number of iterations in a CG method. Our method converges faster than the formulation including pixel values. The error of a kernel estimate is measured by the sum of pixel-wise squared differences from the original kernel.*

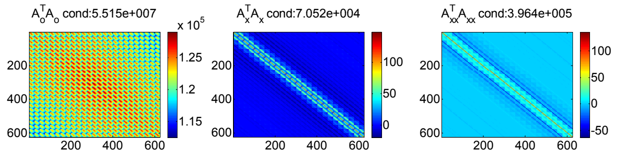

Figure 7: *Visualization of matrix $\mathbf{A}_*^T\mathbf{A}_*$. From left to right, $\mathbf{A}_o^T\mathbf{A}_o$, $\mathbf{A}_x^T\mathbf{A}_x$, and $\mathbf{A}_{xx}^T\mathbf{A}_{xx}$ computed with the example in Fig. 6.*

on i. Image derivatives are usually close to zero except on edge pixels. Hence, for a derivative image P_*, the values in $\mathbf{A}_*^T\mathbf{A}_*$ are large only around the diagonal and become small rapidly for off-diagonal regions. In contrast, if we involve pixel values for kernel estimation, $\mathbf{A}_o^T\mathbf{A}_o$ from the latent image L will be included in the summation of Eq. (11). As pixel values are mostly non-zero over an image, $\mathbf{A}_o^T\mathbf{A}_o$ have large values except for far off-diagonal regions.

Fig. 7 visualizes $\mathbf{A}_o^T\mathbf{A}_o$, $\mathbf{A}_x^T\mathbf{A}_x$, and $\mathbf{A}_{xx}^T\mathbf{A}_{xx}$ computed with the example image in Fig. 6. Condition numbers of the matrices are 5.5155×10^7, 7.0516×10^4 and 3.9636×10^5, respectively. Clearly, $\mathbf{A}_x^T\mathbf{A}_x$, and $\mathbf{A}_{xx}^T\mathbf{A}_{xx}$ have diagonally dominant structures and smaller condition numbers than $\mathbf{A}_o^T\mathbf{A}_o$. As a result, we can reduce the number of iterations in the CG method used for kernel estimation by excluding pixel values in the energy function.

Speed comparison For speed comparison, we consider a kernel estimation method, whose energy function is the same as [Shan et al. 2008], except that the L_2 norm is used for a kernel prior. We call the method Shan-L2. Note that the original version in [Shan et al. 2008] needs more computation, since it uses the L_1 norm for a kernel prior. Also, according to the authors' document on additional programming details, which is available on their website, their kernel estimation step directly assembles huge matrices, which results in memory shortage and excessive computation. While our method needs two FFTs per CG iteration, Shan-L2 needs 14 FFTs, as it uses six images, one pixel value image plus five derivative images. Moreover, since it uses pixel values, it needs more iterations than ours, as analyzed above. In our experiments, our method runs five CG iterations, while Shan-L2 needs about 30 iterations to obtain a similar accuracy. Other previous methods usually use only pixel values for kernel estimation. In this case, convolutions cannot be concatenated due to the boundary problem, so they need four FFTs to compute a gradient of their energy function. They would also need more iterations than ours. To sum up, our kernel estimation is more than 40 times faster than Shan-L2 and more than 10 times faster than other methods using only pixel values.

Additionally, we found that the kernel estimation in [Shan et al. 2008] is more time-consuming than their latent image estimation. Even if Shan-L2 is used instead of the original version, kernel estimation requires about 10 times more FFTs than latent image estimation. Consequently, when we set [Shan et al. 2008] as a baseline, kernel estimation contributes more to acceleration than latent image estimation in our method.

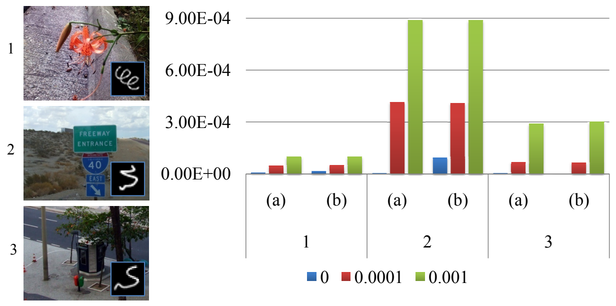

Figure 8: *Kernel estimation error. Left: test images with blur kernels. Right: errors of the estimated kernels with (a) our method and (b) the method including pixel values. Errors are measured by the sum of pixel-wise squared differences between the estimated and original kernels. Different colors denote different amounts of added noise. For all test cases, (a) and (b) show similar errors.*

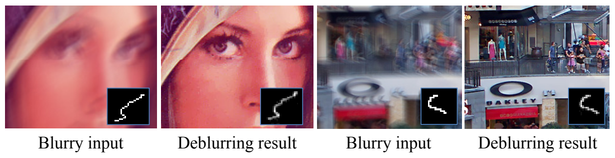

Figure 9: *Deblurring of synthetic examples. The first and third images show blurred images with applied motion blur kernels. The second and fourth images show our deblurring results with estimated kernels. The kernel sizes of the left and right examples are 29×25 and 21×21, respectively.*

Accuracy To test the accuracy of our kernel estimation method, we experimented with synthetic examples, where we added Gaussian noise with different variances. For the examples, we estimated the kernels using our method and the other method including pixel values. Fig. 8 shows the errors of estimated kernels. Although our method does not use pixel values, it exhibits similar accuracy to the case of including pixel values. Furthermore, our method gives better accuracy for some cases due to the well-conditioned system obtained by excluding pixel values.

Application to other methods Our acceleration technique for optimizing Eq. (6) can be used for other energy functions of kernel estimation. If the gradient of an energy function can be represented by a similar form to Eq. (9), the optimization process can be accelerated by our technique. For example, in the kernel estimation of [Yuan et al. 2007], we can use the gradient maps of a noisy image for Eq. (6), instead of $\{P_x, P_y\}$, and accelerate kernel estimation with our technique. Similarly, our acceleration technique can be used for kernel estimation of [Jia 2007; Shan et al. 2008].

6 Results

In our experiments, the kernel size was specified by the user and did not have much influence on the accuracy of kernel estimation if the size was large enough to contain the estimated kernel. Although our method contains several parameters, we mainly controlled the kernel size, the range sigma σ_r of bilateral filtering for prediction, and the parameters of the final deconvolution operation which are adopted from [Shan et al. 2008].

To demonstrate the quality of the estimated motion blur kernels, we first show deblurring results with synthetic examples. In Fig. 9, the estimated motion blur kernels have almost the same shapes as the original. The deblurring results show that the fine details of the original latent images have been accurately recovered.

Figs. 1 and 10 show deblurring results of real photographs. The photographs contain complex structures and different camera motions. In the deblurring results, sharp edges have been significantly enhanced, revealing the object shapes and structures more clearly.

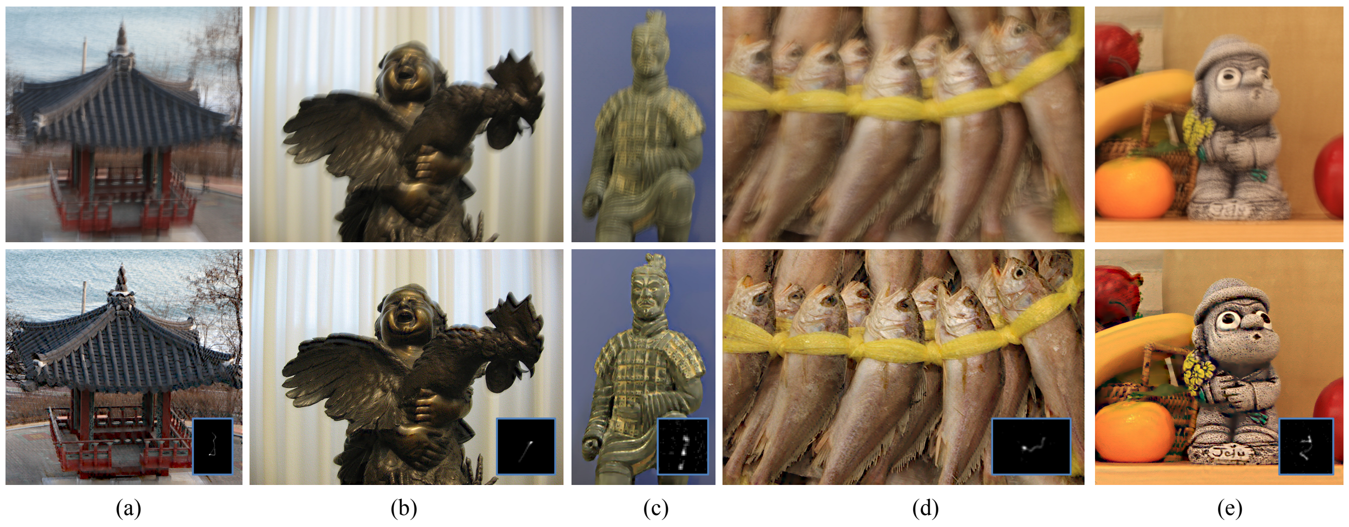

Figure 10: *Deblurring results of real photographs. Top row: input blurred images. Bottom row: deblurring results with estimated kernels.*

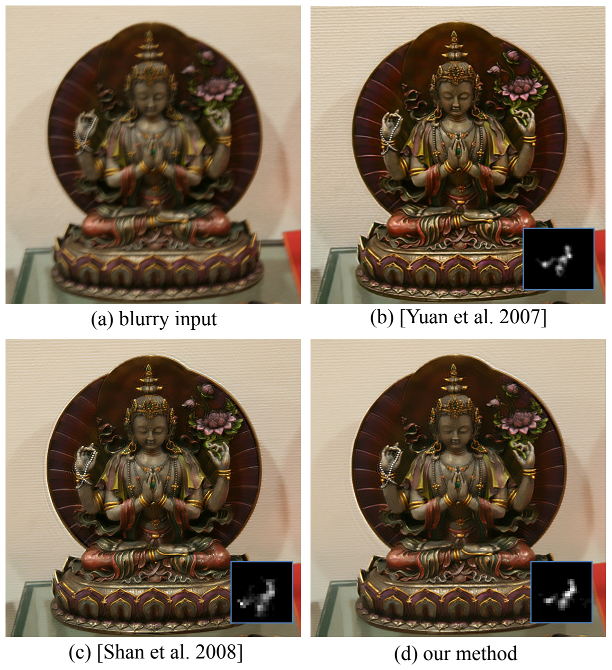

Figure 11: *Comparison with previous deblurring methods.*

The estimated motion blur kernels show reasonable shapes. Additional results can be found in the supplementary material.

Table 1 shows the processing times for the deblurring examples in Fig. 10. We implemented our method with GPU acceleration using BSGP [Hou et al. 2008]. For FFT, we used a FFT library provided with CUDA. Since BSGP is similar to C-language and easy to use, GPU implementation was almost straightforward. Our testing environment was a PC running MS Windows XP 32bit version with Intel Core2 Quad CPU 2.66GHz, 3.25GB RAM, and an NVIDIA GeForce GTX 280 graphics card. Even for kernels of large sizes, our method can remove blurs from input images of moderate sizes within a few seconds. Actually, for the photographs in Figs. 10(c), 10(d), and 10(e), it took less than one second.

Fig. 11 compares our deblurring results with previous methods. A pair of blurred and noisy images is needed in [Yuan et al. 2007] while a given single image can be deblurred in [Shan et al. 2008]. Although our method uses a single image and the computation is much faster than the previous methods, the quality of the results is comparable due to our accurate kernel estimation. Quantitative comparison of our method with previous deblurring methods using the data set of [Levin et al. 2009] can be found in the supplementary material.

Image	Size		Processing time (sec.)		
	Image	Kernel	A	B	C
a	972 × 966	65 × 93	2.188	3.546	5.766
b	1024 × 768	49 × 47	1.062	1.047	2.125
c	483 × 791	35 × 39	0.343	0.235	0.578
d	858 × 558	61 × 43	0.406	0.297	0.703
e	846 × 802	35 × 49	0.516	0.406	0.922

Table 1: *Processing times of the deblurring examples in Fig. 10. A: kernel estimation. B: final deconvolution. C: total processing time.*

Image	Size		Processing time (sec.)		
	Image	Kernel	A	B	C
Picasso	800 × 532	27 × 19	360	20	0.609
statue	903 × 910	25 × 25	762	33	0.984
night	836 × 804	27 × 21	762	28	0.937
red tree	454 × 588	27 × 27	309	11	0.438

Table 2: *Processing time comparison. A: [Shan et al. 2008]. B: our method with C++ implementation. C: our method with GPU acceleration.*

Table 2 compares the processing times of our method and [Shan et al. 2008]. For comparison, we also implemented our method using C++ without GPU acceleration. To measure the computation time of [Shan et al. 2008], we used the executable provided by the authors on the internet[1]. We used the four images included in the internet distribution with the parameters provided by the authors. The processing times of our method include the time for the final deconvolution, which is performed by our C++ or GPU implementation of the method in [Shan et al. 2008]. To exclude additional deconvolution time from the comparison, we did not use the additional ringing suppression method of [Shan et al. 2008] for both their and our results. Table 2 shows that with a C++ implementation, our deblurring method is about 20 times faster than the executable of [Shan et al. 2008], which is a C-language implementation.

7 Discussion

In this paper, we proposed a fast blind deconvolution method, which provides enough speed for consumers to use deblurring in practice.

[1] http://www.cse.cuhk.edu.hk/~leojia/projects/motion_deblurring/index.html

Our method is based on the intuition that blurred strong edges can provide reliable information for the faithful estimation of a motion blur kernel. With the intuition, we could specify the required properties of an estimated latent image used for kernel estimation in an iterative deblurring process. By explicitly achieving the properties with simple image filters in a multi-scale approach, we could avoid using computationally expensive priors for handling complex and large blurs. We hope that this intuition would help in solving more difficult deblurring problems, such as spatially-varying motion blurs.

Limitations Our deblurring method consists of simpler steps than previous methods. This simplicity may incur degradation of the deblurring quality, although the comparisons in Fig. 11 and the supplementary material show comparable quality to previous sophisticated methods. Our prediction depends on local features rather than global structures of the image. If an image has strong local features inconsistent with other image regions, our method may fail to find a globally optimal solution. For example, such inconsistent features can be caused by saturated pixels, as shown in the supplementary material. Our kernel estimation uses the Tikhonov regularization term, which is simpler than a sparsity prior used in [Fergus et al. 2006]. Although less accurate kernels could be obtained with this simple term, the experiments with synthetic examples in Fig. 9 and the supplementary material demonstrate that the accuracy of our kernel estimation is reasonable.

We observed that our deblurring results are relatively sensitive to parameters, and improving the robustness to the parameters is our future work. Nevertheless, we could obtain desirable results usually within a few trials, where even several trials need much less time than a single run of previous methods.

We assume the latent image contains enough sharp edges that can be used for kernel estimation. While most photographs include people, buildings, and natural scenery, which usually have sharp edges, this assumption may not always hold, e.g., with a photograph of furry objects. In this case, kernel estimation would be less accurate and the deblurring result could be over-sharpened.

Our method shares common limitations with other uniform motion deblurring methods. Due to the limitation of the blur model based on convolution, saturated pixels from strong lights, severe noise, and a spatially varying blur would not be properly handled. Extending our deblurring method to resolve this limitation will be interesting future work.

Acknowledgements We thank Flickr users, tanakawho (Fig. 4) and AndWat (Fig. 10(b)), and Nae-Jin Kong (Fig. 9 right) for sharing their images. This work was supported by the IT R&D program of MKE/MCST/KEIT (2008-F-031-01, Development of Computational Photography Technologies for Image and Video Contents) and the ERC Program of MEST/NRF (R11-2008-007-01002-3).

References

BEN-EZRA, M., AND NAYAR, S. K. 2004. Motion-based motion deblurring. *IEEE Trans. Pattern Analysis Machine Intelligence 26*, 6, 689–698.

CHAN, T. F., AND WONG, C.-K. 1998. Total variation blind deconvolution. *IEEE Trans. Image Processing 7*, 3, 370–375.

CHEN, W.-G., NANDHAKUMAR, N., AND MARTIN, W. N. 1996. Image motion estimation from motion smear - a new computational model. *IEEE Trans. Pattern Analysis Machine Intelligence 18*, 4, 412–425.

CHO, S., MATSUSHITA, Y., AND LEE, S. 2007. Removing non-uniform motion blur from images. In *Proc. ICCV 2007*, 1–8.

DAI, S., AND WU, Y. 2008. Motion from blur. In *Proc. CVPR 2008*, 1–8.

FERGUS, R., SINGH, B., HERTZMANN, A., ROWEIS, S. T., AND FREEMAN, W. 2006. Removing camera shake from a single photograph. *ACM Trans. Graphics 25*, 3, 787–794.

HOU, Q., ZHOU, K., AND GUO, B. 2008. BSGP: bulk-synchronous GPU programming. *ACM Trans. Graphics 27*, 3, article no. 19.

JI, H., AND LIU, C. 2008. Motion blur identification from image gradients. In *Proc. CVPR 2008*, 1–8.

JIA, J. 2007. Single image motion deblurring using transparency. In *Proc. CVPR 2007*, 1–8.

JOSHI, N., SZELISKI, R., AND KREIGMAN, D. 2008. PSF estimation using sharp edge prediction. In *Proc. CVPR 2008*, 1–8.

JOSHI, N. 2008. *Enhancing photographs using content-specific image priors*. PhD thesis, UCSD.

LEVIN, A., FERGUS, R., DURAND, F., AND FREEMAN, W. T. 2007. Image and depth from a conventional camera with a coded aperture. *ACM Trans. Graphics 26*, 3, article no. 70.

LEVIN, A., SAND, P., CHO, T. S., DURAND, F., AND FREEMAN, W. T. 2008. Motion-invariant photography. *ACM Trans. Graphics 27*, 3, article no. 71.

LEVIN, A., WEISS, Y., DURAND, F., AND FREEMAN, W. 2009. Understanding and evaluating blind deconvolution algorithms. In *Proc. CVPR 2009*, 1–8.

LUCY, L. 1974. An iterative technique for the rectification of observed distributions. *Astronomical Journal 79*, 6, 745–754.

MONEY, J. H., AND KANG, S. H. 2008. Total variation minimizing blind deconvolution with shock filter reference. *Image and Vision Computing 26*, 2, 302–314.

OSHER, S., AND RUDIN, L. I. 1990. Feature-oriented image enhancement using shock filters. *SIAM Journal on Numerical Analysis 27*, 4, 919–940.

RAV-ACHA, A., AND PELEG, S. 2005. Two motion-blurred images are better than one. *Pattern Recognition Letters 26*, 311–317.

SHAN, Q., JIA, J., AND AGARWALA, A. 2008. High-quality motion deblurring from a single image. *ACM Trans. Graphics 27*, 3, article no. 73.

TAI, Y.-W., DU, H., BROWN, M. S., AND LIN, S. 2008. Image/video deblurring using a hybrid camera. In *Proc. CVPR 2008*, 1–8.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *Proc. ICCV 1998*, 839–846.

WIENER, N. 1964. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. MIT Press.

YITZHAKY, Y., MOR, I., LANTZMAN, A., AND KOPEIKA, N. S. 1998. Direct method for restoration of motion-blurred images. *Journal of Opt. Soc. Am. A. 15*, 6, 1512–1519.

YUAN, L., SUN, J., QUAN, L., AND SHUM, H.-Y. 2007. Image deblurring with blurred/noisy image pairs. *ACM Trans. Graphics 26*, 3, article no. 1.

YUAN, L., SUN, J., QUAN, L., AND SHUM, H.-Y. 2008. Progressive inter-scale and intra-scale non-blind image deconvolution. *ACM Trans. Graphics 27*, 3, article no. 74.