# Stochastic Blind Motion Deblurring

Lei Xiao, James Gregson, Felix Heide, and Wolfgang Heidrich

***Abstract*— Blind motion deblurring from a single image is a highly under-constrained problem with many degenerate solutions. A good approximation of the intrinsic image can, therefore, only be obtained with the help of prior information in the form of (often nonconvex) regularization terms for both the intrinsic image and the kernel. While the best choice of image priors is still a topic of ongoing investigation, this research is made more complicated by the fact that historically each new prior requires the development of a custom optimization method. In this paper, we develop a stochastic optimization method for blind deconvolution. Since this stochastic solver does not require the explicit computation of the gradient of the objective function and uses only efficient local evaluation of the objective, new priors can be implemented and tested very quickly. We demonstrate that this framework, in combination with different image priors produces results with Peak Signal-to-Noise Ratio (PSNR) values that match or exceed the results obtained by much more complex state-of-the-art blind motion deblurring algorithms.**

***Index Terms*— Motion deblur, blind deconvolution, stochastic random walk, cross channel prior, chromatic kernel, saturated pixels, Poisson noise.**

## I. Introduction

The goal of deconvolution is to recover a sharp intrinsic image $\mathbf{I}$ from a blurred image $\mathbf{B}$. The image formation process can be modeled as

$$\mathbf{B} = \mathbf{K} \otimes \mathbf{I} + \mathbf{N}, \tag{1}$$

where $\mathbf{K}$ represents the blur kernel, $\otimes$ represents discrete convolution, and $\mathbf{N}$ is a noise term. We first assume Gaussian noise for simplicity, and extend our algorithm for Poisson noise in Section IV-D.

While there are many applications in which the kernel $\mathbf{K}$ is known or can be calibrated a priori (e.g. aberration correction) or is considered to be from a small set of possible kernels (depth of field blur), in other situations the kernel is unknown since it is generated by a process that cannot be replicated (e.g. object motion or camera shake).

This *blind* case, where both the kernel and the intrinsic image is unknown is highly under-constrained, and is subject to many degenerate solutions, including one where the estimated kernel is simply a Dirac peak. It is therefore obvious that blind deconvolution is only feasible with the use of additional information, either in the form of additional sensor data (e.g. inertial sensors [1] or multiple input images of the same scene [2]), or in the form of prior information (image priors). While good priors for both the images and the blur kernels are still an active area of investigation, many choices that have been proposed are non-linear and often even non-convex. This makes it difficult and time consuming to experiment with different image priors, since each new candidate typically requires a customized optimization procedure that can require a significant effort to implement.

In this paper we extend our recent work in stochastic *non-blind* deconvolution [3] to derive a *blind* method that is purely based on stochastic sampling. The method relies entirely on local evaluations of the objective function, without the need to compute gradients. This makes it effortless to implement and test new image priors. In our implementation we focus on spatially-invariant blur kernels, although extensions such as tile-based kernels would be straightforward to add. Specifically we present the following contributions:

- a stochastic framework for blind deconvolution,
- a demonstration of stochastic optimization for non-convex priors for the image (e.g. sparse derivatives and cross-channel information), the kernel (e.g. anisotropic diffusion), and even the data term (handling of saturation and clamping, as well as an Anscombe transformation for Poisson noise).
- an implementation with a range of sparsity and color priors for the image, as well as sparsity and smoothness priors for the kernel, that together match or outperform existing state-of-the art methods,
- a method for recovering colored blur kernels that arrive, e.g. in remote sensing where the exposure time varies per channel (see [4]), and
- a method for dealing with clipped color values in non-blind deconvolution.

The rest of this paper is organized as follows. In Section II, we briefly review the previous work of single-image blind deconvolution and describe the stochastic optimization method. In Section III and IV, we present our approach and describe its critical implementation details. In Section V, we show our results of both synthetic images

Manuscript received March 28, 2014; revised August 29, 2014, December 8, 2014, and March 24, 2015; accepted May 4, 2015. Date of publication May 13, 2015; date of current version June 9, 2015. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada through the Discovery Grant, in part by the University of British Columbia Four Year Fellowship, and in part by the Baseline Funding through the King Abdullah University of Science and Technology, Thuwal, Saudi Arabia. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Alessandro Foi.

L. Xiao, J. Gregson, and F. Heide are with the Department of Computer Science, The University of British Columbia, Vancouver, BC V6T 1Z4, Canada (e-mail: leixiao@cs.ubc.ca; jgregson@cs.ubc.ca; fheide@cs.ubc.ca).

W. Heidrich is with the Visual Computing Center, King Abdullah University of Science and Technology, Thuwal 23955-6900, Saudi Arabia (e-mail: wolfgang.heidrich@kaust.edu.sa).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org., provided by the author. The file contains PSNR comparisons of benchmark dataset, result images of benchmark dataset, result images of running-time analysis, and result images of noise influence analysis. The total size of the file is 43.5 MB. Contact leixiao@cs.ubc.ca for further information.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2015.2432716

and real photographs. In Section VI, we conclude this paper and discuss potential further work.

## II. Related Work

### A. Known Point-Spread Function

In cases with a known blur kernel, the method is presented with one or more blurry inputs and used to estimate sharp intrinsic images. Application-specific kernels may be obtained in a variety of ways, either by explicit design [5], [6], inertial measurement [1], [7], calibration [8], [9] or known from capture conditions. Kernels may also be estimated.

A key distinguishing feature of non-blind methods are the priors that are applied. $l_1$ priors enforce solution sparsity see [10]–[12], with efficient solvers and well-developed theory. Natural image statistics (e.g., heavy-tailed gradient distribution) are also popular, see [6], [13], but are non-convex and often difficult to solve. More recently, non-local filters have been used for image denoising, see [14]–[16], and are starting to be adapted to deblurring [17], [18].

### B. Unknown Point-Spread Function

When the blur kernel is not readily available, the problem becomes much more challenging especially for single image input. Recent success arises from the use of sparse priors and multi-scale scheme. Fergus *et al.* [13] fits the heavy-tailed prior by a mixture of Gaussians and solves the intrinsic image gradient and blur kernel by a variational Bayesian method [19]. Richardson-Lucy algorithm [20], [21] is then applied to reconstruct the final intrinsic image with the estimated kernel. Shan *et al.* [22] introduced a model of the spatial randomness of noise and a local smoothness prior, and estimates the intrinsic image and kernel in a unified framework. Krishnan *et al.* [23] introduced a scale-invariant $l_1/l_2$ prior which compensates for the attenuation of high frequencies in the blurry image. Xu *et al.* [24] proposed to use the $l_0$ regularizer on the image gradient at intermediate steps of blind kernel estimation and solved the optimization by a proximal algorithm [25]. Pan *et al.* [26] used $l_0$ regularized intensity and gradient priors for text deblurring. Customized optimization methods were used for specific priors in these algorithms. While developing better priors is still a promising direction for future research, we believe our framework which can allow for quick exploration of new ideas in this space can be valuable.

Another type of methods use filtering approaches to extract strong image edges from which kernels may be estimated rapidly. Cho and Lee [27] adopted shock filter [28] and bilateral filter [29] to predict sharp edges. Xu and Jia [30] proposed a edge selection process based on the observation that strong edges do not always profit kernel estimation. They also proposed a kernel refinement method by iterative support detection [31]. These algorithms cannot capture the sparsity of the image and kernel and can result in noisy estimates. The use of shock filter and bilateral filter can result in oversharpened edges and halo artifacts.

In case of highly noisy input, Tai and Lin [32] proposed a method for jointly denoising and deblurring the image. Zhong *et al.* [33] applied directional low-pass filters at different orientations to the input image and estimated the Radon transform of the blur kernel from each filtered image, while the final blur kernel is computed by inverse Radon transform.

**Algorithm 1** Stochastic Blind Deconvolution Framework

```
Input: Blurry image $\mathbf{B}$; weights for priors on intrinsic image: $\theta_{\mathbf{I}}$; weights for priors on the kernel: $\theta_{\mathbf{K}}$; maximum blur kernel size $\phi$; number of iterations $T$.
Output: Estimated intrinsic image $\hat{\mathbf{I}}$; estimated kernel $\hat{\mathbf{K}}$.
 1: $S = \lceil 2\log_2(\phi/3) + 1 \rceil$ //number of scales
 2: for $s = 1$ to $S$ do
 3:     $\mathbf{B}^s = downsample(\mathbf{B}, s, 'bilinear')$
 4:     if $s == 1$ then
 5:         $\hat{\mathbf{I}}^s = \mathbf{B}^s$
 6:         $\hat{\mathbf{K}}^s = initializeKernel(\mathbf{B})$
 7:     else
 8:         $\hat{\mathbf{I}}^s = upsample(\hat{\mathbf{I}}^{s-1}, \sqrt{2}, 'bicubic')$
 9:         $\hat{\mathbf{K}}^s = upsample(\hat{\mathbf{K}}^{s-1}, \sqrt{2}, 'nearestneighbor')$
10:     end if
11:     for $t = 1$ to $T$ do
12:         $\hat{\mathbf{I}}^s = updateIntrinsicImage(\mathbf{B}^s, \hat{\mathbf{K}}^s, \hat{\mathbf{I}}^s, \theta_{\mathbf{I}})$
13:         $\hat{\mathbf{K}}^s = updateKernel(\mathbf{B}^s, \hat{\mathbf{I}}^s, \hat{\mathbf{K}}^s, \theta_{\mathbf{K}})$
14:     end for
15:     $[\theta_{\mathbf{I}}, \theta_{\mathbf{K}}] = updateWeights(\theta_{\mathbf{I}}, \theta_{\mathbf{K}})$
16: end for
17: $\hat{\mathbf{K}} = \hat{\mathbf{K}}^S$ //final estimated kernel
18: $\hat{\mathbf{I}} = updateIntrinsicImage(\mathbf{B}, \hat{\mathbf{K}}, \hat{\mathbf{I}}^S, \theta_{\mathbf{I}})$ //final intrinsic image restoration
```

For non-uniform blur due to camera rotation, Whyte *et al.* [34] proposed a parameterized geometric model of the blurring process considering the rotational velocity of the camera during exposure. Gupta *et al.* [35] modeled the spatially varying kernels by a motion density function which records the fraction of time spent in each discretized portion of the space by the camera during exposure. Harmeling *et al.* [36] and Hirsch *et al.* [37] combined the global camera motion model and local patch uniform deblurring to accelerate the non-uniform kernel estimation.

### C. Stochastic Random Walk Optimization Algorithm

In earlier work, we presented stochastic random-walk optimization for tomography [38] and non-blind deblurring [3] that uses many incremental local solution updates at sampled locations. Sample placement is driven by a stochastic random walk that favors local exploration where fast progress has recently been made. The advantage of this approach is a simple implementation and straightforward inclusion of complex priors, at the cost of missing theoretical convergence guarantees for non-smooth objectives, as well as a runtime penalty. In this work, we extend the approach to handle blind deconvolution.

## III. Basic Algorithm

Our algorithm is based on a multiscale approach that iterates between kernel estimation and solving a non-blind deconvolution step (Algorithm 1). The algorithm uses a scale

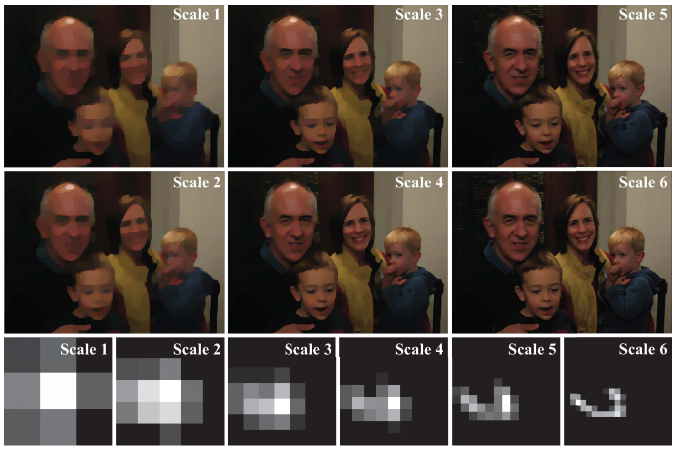


Fig. 1. Our multi-scale scheme on the scene shown in Fig. 8. *Row 1 & 2*: intrinsic images estimated at each scale. *Row 3*: kernel estimated at each scale. In contrast to most existing blind deconvolution methods, our algorithm can recover all color channels of the intrinsic image simultaneously using the cross-channel information.

space to avoid local optima, working its way from the coarse scales to the fine scales (also see Fig. 1). At each scale, the method upsamples the kernel and intrinsic image from the next coarser scale using nearest-neighbor and bicubic upsampling respectively and then alternately updates the current estimates for the intrinsic image (Section III-A) and the kernel (Section III-B) at the current scale in an inner loop. Next, prior weights are adjusted (Section III-C) before moving to the next finer scale. At the coarsest scale, the kernel is initialized as a 3 by 3 image with either a horizontal or vertical stripe depending on the dominant gradient direction [13], while the blurry image for each scale is obtained with bilinear downsampling from the full-resolution blurry image.

## A. Updating the Intrinsic Image $\hat{\mathbf{I}}^s$

Given the kernel estimated at the current scale $\hat{\mathbf{K}}^s$, the function *updateIntrinsicImage*() in Algorithm 1 updates our estimate of the intrinsic image by solving a non-blind deconvolution problem using a stochastic random walk optimization (Algorithm 2, also see [3]), which minimizes objectives of the form:

$$f(\hat{\mathbf{I}}^s) = ||\mathbf{B}^s - \hat{\mathbf{K}}^s \otimes \hat{\mathbf{I}}^s||_2^2 + \theta_{\mathbf{I}} \cdot g(\hat{\mathbf{I}}^s) \tag{2}$$

The quadratic term gives the data-fitting error. $g(.) = [g_1(.), \dots, g_R(.)]^T$ is a vector of individual regularizers, and $\theta_{\mathbf{I}} = [\theta_{\mathbf{I}1}, \dots, \theta_{\mathbf{I}R}]^T$ is the corresponding vector of weights for each regularization term. The total weighted penalty (scalar) is the dot-product of $\theta_{\mathbf{I}}$ and $g(.)$. Examples of $g_i(.)$ are given in Eq. 8 - 12.

*1) Random Walk Process:* As summarized in Algorithm 2, we create a random walk chain of pixel location $\mathbf{x}_i$ in the support domain of the intrinsic image at which we propose to add to or remove from an energy quantum $ed_I$:

$$\hat{\mathbf{I}}_i^s = \hat{\mathbf{I}}_{i-1}^s \pm ed_I \cdot \delta_{\mathbf{x}_i}, \tag{3}$$

where $\delta_{\mathbf{x}_i}$ is the characteristic function (i.e., Kronecker delta function) for the sample pixel $\mathbf{x}_i$, and $\hat{\mathbf{I}}_i^s$ the estimated intrinsic

**Algorithm 2** Stochastic Random Walk Optimization

**Input:** $\hat{\mathbf{I}}_0$ as the initial value of the signal $\hat{\mathbf{I}}$ to optimize, with support domain $\Omega$; function $c(\mathbf{x}_i)$ evaluating the change of objective given sample $\mathbf{x}_i$; function $t(\mathbf{x}_i|\mathbf{x}_{i-1})$ mutating sample location from $\mathbf{x}_{i-1}$ to $\mathbf{x}_i$; $ed$ as the initial magnitude of the sample energy; $\epsilon$ as a constant scalar (by default 0.001); $\gamma$ as a constant scalar between 0 and 1 (by default 0.1); $N$ as the number of iterations; $M$ as the total number of proposed samples in each iteration.

**Output:** Updated signal $\hat{\mathbf{I}}$.

```
1:  $\mathbf{x}_0 \leftarrow random(\Omega)$ //initialize the random walk chain by uniform-randomly sampling the support domain $\Omega$.
2:  for $j = 1$ to $N$ do
3:      $\beta \leftarrow 0$ //number of accepted samples in each iteration
4:      for $i = 1$ to $M$ do
5:          $\mathbf{x}_i \leftarrow sample(\mathbf{x}_{i-1}, t(\mathbf{x}_i|\mathbf{x}_{i-1}), ed)$ //propose a new sample $\mathbf{x}_i$ with energy $ed$ given the previous sample $\mathbf{x}_{i-1}$ and transition function $t(.)$.
6:          if $c(\mathbf{x}_i) < 0$ then
7:              $\beta \leftarrow \beta + 1$
8:              $accept(\mathbf{x}_i)$ //if the new sample $\mathbf{x}_i$ decreases the objective, accept it and update $\hat{\mathbf{I}}$ (e.g., by Eq. 3).
9:          end if
10:         $a \leftarrow c(\mathbf{x}_i)/c(\mathbf{x}_{i-1})$ //compute the rate of objective change given the new sample $\mathbf{x}_i$.
11:         if $a < random([0,1])$ and $c(\mathbf{x}_{i-1}) < 0$ then
12:             $\mathbf{x}_i \leftarrow \mathbf{x}_{i-1}$ //if $a$ is lower than a uniformly distributed random real number between 0 and 1, and the previous sample $\mathbf{x}_{i-1}$ decreased the objective, keep the random walk chain at previous sample location $\mathbf{x}_{i-1}$; otherwise, update the random walk chain to $\mathbf{x}_i$.
13:         end if
14:     end for
15:     $\mathbf{x}_0 \leftarrow \mathbf{x}_M$ //update the root of random walk chain for next iteration.
16:     if $\beta < \epsilon \cdot M$ and $ed < \epsilon$ then
17:         break; //if too few samples were accepted and $ed$ is too small meanwhile, stop the iterations.
18:     else if $\beta < \gamma \cdot M$ then
19:         $ed \leftarrow 0.5 \cdot ed$ //reduce the sample magnitude by half when too few samples were accepted.
20:     end if
21: end for
```

image at $i^{th}$ iteration of the random walk. Both the positive and negative energy are evaluated at the sample pixel but only the sample that decreases the objective function most is kept.

The quantity $c(\mathbf{x}_i)$ measures the change of the objective function $f(.)$ if the proposed sample $\mathbf{x}_i$ with value $\pm ed_I$ were to be accepted, i.e.,

$$c(\mathbf{x}_i) = f(\hat{\mathbf{I}}_i^s) - f(\hat{\mathbf{I}}_{i-1}^s) \tag{4}$$

If the sample decreases the objective (i.e. $c(\mathbf{x}_i) < 0$), it is accepted and Eq. 3 is applied (lines 6-9, Algorithm 2).

The quantity $a$ measures the rate of objective change given the new proposed sample $\mathbf{x}_i$, as defined in Eq. 5.

$$a = c(\mathbf{x}_i)/c(\mathbf{x}_{i-1}) \tag{5}$$

If $a$ is lower than a uniform-randomly generated real number between 0 and 1 (a.k.a *Russian roulette* strategy in Metropolis-Hasting sampling techniques [39]), and the previous sample $\mathbf{x}_{i-1}$ decreased the objective, the new sample is discarded entirely leaving the intrinsic image and random walk chain unchanged (lines 11-13, Algorithm 2). Otherwise, the random walk chain is updated to the new sample location $\mathbf{x}_i$.

*2) Evaluating $c(\mathbf{x}_i)$:* Given a proposed sample, the objective function in Eq. 2 changes locally because of the compact support of the blur kernel $\hat{\mathbf{K}}^s$ and priors $g(.)$. We can efficiently evaluate $c(\mathbf{x}_i)$ by only considering the local neighborhood of the given sample pixel.

In practice, we keep a second sequence of images $\hat{\mathbf{B}}_i^s = \hat{\mathbf{K}}^s \otimes \hat{\mathbf{I}}_i^s$, which represents the blurred image we would expect if the intrinsic image was $\hat{\mathbf{I}}_i^s$. The $\hat{\mathbf{B}}_i^s$ can be updated efficiently by splatting $\hat{\mathbf{K}}^s \otimes \delta_{\mathbf{x}_i}$ during the random walk process:

$$\hat{\mathbf{B}}_i^s = \hat{\mathbf{B}}_{i-1}^s \pm ed_I(\hat{\mathbf{K}}^s \otimes \delta_{\mathbf{x}_i}) \tag{6}$$

The splat $\hat{\mathbf{K}}^s \otimes \delta_{\mathbf{x}_i}$ is just a shifted, and mirrored copy of the kernel $\hat{\mathbf{K}}^s$ and is pre-computed for acceleration. The change in the regularization term is evaluated in an analogous manner but is specific to chosen regularizers.

*3) Sample Mutation:* The function $sample(.)$ generates a new sample $\mathbf{x}_i$ from the previous sample $\mathbf{x}_{i-1}$ by the mutation function $t(\mathbf{x}_i|\mathbf{x}_{i-1})$. Two types of mutation strategies are used.

The first strategy generates $\mathbf{x}_i$ by a zero-mean Gaussian-distributed random offset $\eta(0, \sigma)$ from $\mathbf{x}_{i-1}$. This mutation allows more samples to be drawn in the regions where they can reduce the objective function more effectively. Using a Gaussian distribution ensures ergodicity of the random walk process. The standard deviation of the Gaussian kernel $\sigma$ is set to 4 pixels when updating intrinsic image.

The second mutation strategy chooses the new sample randomly in the support domain with uniform probability. We stimulate this mutation with 2% probability during the random walk process. This helps to avoid start-up bias and also contributes to ergodicity of the random walk process.

Eq. 7 gives the formula of the above mutation strategies:

$$\mathbf{x}_i = \begin{cases} random(\Omega), & \text{if } q < 0.02; \\ \mathbf{x}_{i-1} + \eta(0, \sigma), & \text{otherwise} \end{cases} \tag{7}$$

where $random(\Omega)$ means a random pixel across the image support domain $\Omega$, and $q$ is a random real number between 0 and 1 generated online at each mutation.

*4) Sample Energy:* The magnitude of the sample energy $ed_I$ is reset to be an initial large value at the beginning of each scale, and adjusted iteratively at each scale. It is reduced by half whenever the percentage of accepted samples over all proposed ones in previous iteration goes below a constant scalar $\gamma \in [0, 1]$. This allows the method to take large steps early and make more subtle changes as the minimization proceeds. In our experiments we use the initial value of $ed_I$ as 0.02 and $\gamma$ as 0.1. Note that the blurry and intrinsic images are normalized to be between 0 and 1.

*5) Stopping Criteria:* The random walk process is terminated if the percentage of accepted samples in previous iteration goes below a constant threshold $\epsilon$ and the magnitude of the sample energy $ed_I$ is smaller than $\epsilon$ meanwhile (lines 16-20, Algorithm 2). In our experiments we set $\epsilon$ to 0.001.

*6) Regularizers $g(.)$:* A benefit of the stochastic optimization framework is that it allows very general priors to be used with no change to the overall algorithm. We have used a selection of well-known and frequently used regularization terms listed below.

- Anisotropic total variation [12] (convex, but non-smooth):

$$||\nabla_x \hat{\mathbf{I}}^s||_1 + ||\nabla_y \hat{\mathbf{I}}^s||_1 \tag{8}$$

- Isotropic total variation [12] (convex, but non-smooth):

$$||(|\nabla_x \hat{\mathbf{I}}^s|^2 + |\nabla_y \hat{\mathbf{I}}^s|^2)^{1/2}||_1 \tag{9}$$

- Sparse first-order derivatives [6], [13] (non-convex):

$$||\nabla_x \hat{\mathbf{I}}^s||_p + ||\nabla_y \hat{\mathbf{I}}^s||_p, \quad p \in (0, 1] \tag{10}$$

- Sparse gradient [6], [13] (non-convex):

$$||(|\nabla_x \hat{\mathbf{I}}^s|^2 + |\nabla_y \hat{\mathbf{I}}^s|^2)^{1/2}||_p, \quad p \in (0, 1] \tag{11}$$

- Sparse second-order derivatives [40] (non-convex):

$$||\nabla_{xx} \hat{\mathbf{I}}^s||_p + 2||\nabla_{xy} \hat{\mathbf{I}}^s||_p + ||\nabla_{yy} \hat{\mathbf{I}}^s||_p, \quad p \in (0, 1] \tag{12}$$

In addition we use other priors, including one to reduce chromatic artifacts (see Section IV-A) as well as non-convex data term, e.g. in the case of images with Poisson noise (Section IV-D).

*7) Empirical Convergence:* Following the analysis in [3], our stochastic random walk framework is a form of stochastic coordinate descent (SCD, [41]–[43]). In each iteration, the algorithm picks a single pixel in the image and checks if the objective can be reduced by depositing energy in this pixel. This corresponds to picking a single degree-of-freedom (i.e. a single coordinate axis in the vector of unknowns) and descending along that direction, without computing full gradient of the objective function. The difference to other SCD methods is that our algorithm uses the random walk process to exploit spatial coherence in the deconvolution problem and focus the computational effort on regions with sharp edges, where most work is to be done in deconvolution (see Fig. 5).

For smooth objectives, SCD methods provably converge as long as there is a finite probability of choosing each possible coordinate axis. This is ensured by the ergodicity of our mutation strategy. For general, non-smooth objectives no such proof exists (see details in [3]), but in Fig. 3, we show empirical convergence experiments for our method in the case of non-smooth and non-convex objectives in non-blind intrinsic image estimations.

In Fig. 4, we visualize the residual between the true intrinsic and our estimation in selected iterations for the example

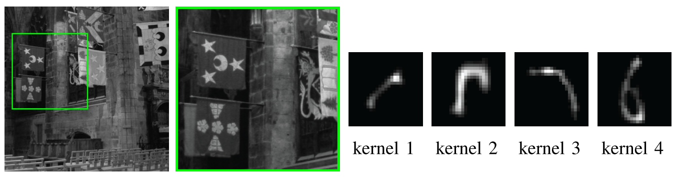


Fig. 2. Ground truth intrinsic image (800×800 pixels) and kernels (25×25 pixels) used for empirical convergence test in Fig. 3 and 6. The kernels are upsampled by nearest neighbor for visualization.

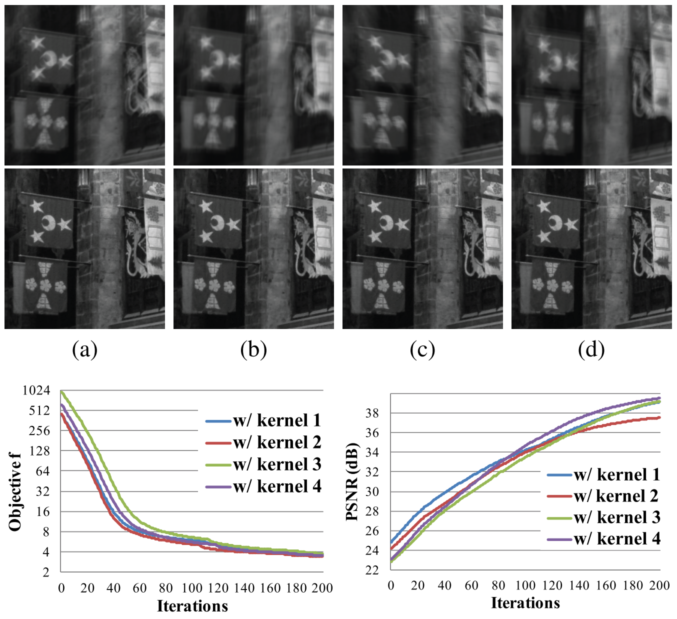


Fig. 3. Example of *non-blind* intrinsic image estimation. In (a)-(d), the 1st row shows the inset of blurry input simulated with the true intrinsic image and each kernel given in Fig. 2, and the 2nd row shows the inset of non-blind estimated intrinsic image by our stochastic random walk method. Only insets are shown due to limited space. The bottom row show the change of objective function $f$ and PSNR when the number of iterations increases. In each iteration, 50000 samples were proposed. Sparse gradient prior (Eq. 11 with p=0.8, *non-convex*) was applied. (a) With kernel 1. (b) With kernel 2. (c) With kernel 3. (d) With kernel 4.

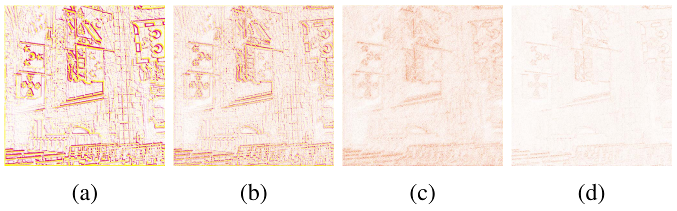


Fig. 4. Visualization of the residual between the ground truth and our estimated intrinsic image at different iterations, for the example shown in Fig. 3(a). The values of the residual are 10× magnified before coded in CMYK colorspace, where *magenta* channel indicates *positive* values and *yellow negative*. (a) Initial. (b) Iteration 20. (c) Iteration 60. (d) Iteration 200.

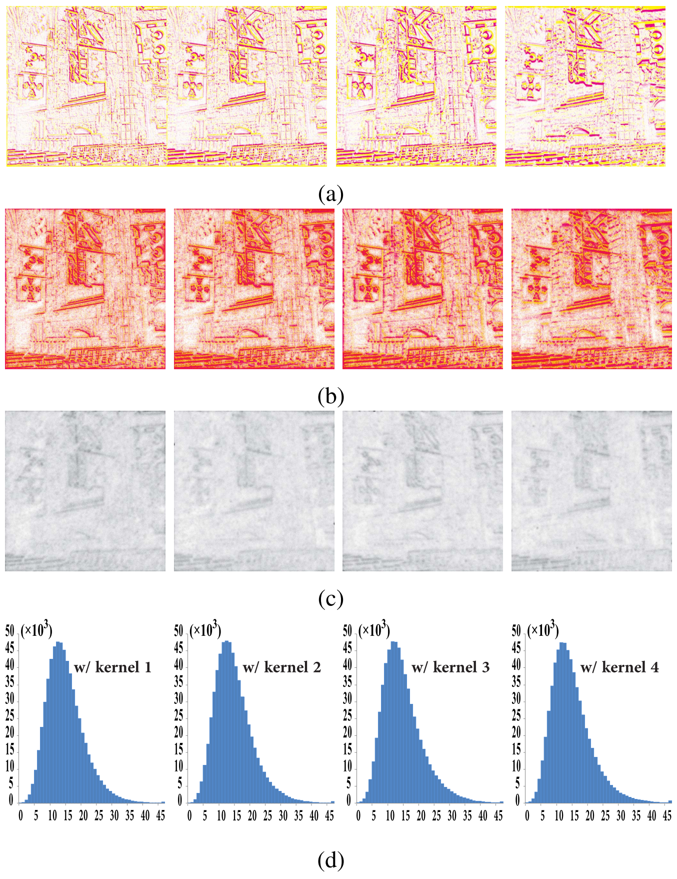


Fig. 5. Visualization of sample distribution for the non-blind examples in Fig. 3. From left to right, the columns show the example with kernel 1, 2, 3, 4. (a) Shows the residual between initial intrinsic image and ground truth. (b) Shows the map of *accepted* sample energy $ed_I$. The values of the residuals and sample energy in (a)-(b) are 10× magnified before coded in CMYK colorspace, where *magenta* channel indicates *positive* values and *yellow negative*. (c) Shows the normalized distribution of proposed sample locations $\mathbf{x}_i$ (including both *accepted* and *rejected* samples.), coded in *key* channel in CMYK colorspace. (d) Shows the histogram of the number of proposed samples in (c). The horizontal axis indicates the bins of the number of proposed samples (clamped at 45 for limited space), and vertical axis indicates the number of pixels at each bin. Note the numbers shown at the vertical axis are in unit of $10^3$. As shown, the majority of pixels consume about 5 to 30 samples.

in Fig. 3(a). The algorithm reduces the residual progressively. In Fig. 5, we visualize the sample distribution in the random walk process for the examples in Fig. 3. As shown in Fig. 5(a), the residual is large mostly at pixels near the image edges. Fig. 5(b) shows the distribution of *accepted* sample energy $ed_I$, which are mostly located at pixels where the residual is large. Note that the positive and negative $ed_I$ partially overlap during the random walk process. Fig. 5(c) shows the distribution of the *number* of proposed samples (including both *accepted* and *rejected* ones). More samples are proposed near the image edges. Fig. 5(d) shows the histogram of the number of proposed samples in Fig. 5(c). The majority of pixels consume 5-30 samples.

The intuitive reasons why such an apparently small amount of samples are required are: 1) the samples are not uniform-randomly proposed. The algorithm uses random walk to exploit the spatial coherence in the images, thus enforce importance sampling; 2) the sample energy $ed$ is initialized as a large value to reduce the number of required samples at the beginning (as larger $ed$ can reduce the objective more effectively), and then progressively reduced to better recover smaller details in the image. This multi-weight strategy help reduce the total number of required samples; 3) each proposed sample is evaluated with both positive and negative energy.

### B. Updating the Kernel $\hat{\mathbf{K}}^s$

The $updateKernel()$ function is used to perform kernel estimation, i.e. to find the kernel that best explains the blurry input $\mathbf{B}^s$ and intrinsic image $\hat{\mathbf{I}}^s$ for a given scale $s$. This operation is performed in derivative domain, minimizing the objective function in Eq. 13 consisting of a data term and set of priors $g(.)$ with weights $\theta_{\mathbf{K}}$:

$$f(\hat{\mathbf{K}}^s) = ||\nabla_{x,y}\mathbf{B}^s - \hat{\mathbf{K}}^s \otimes \nabla_{x,y}\hat{\mathbf{I}}^s||_2^2 + \theta_{\mathbf{K}} \cdot g(\hat{\mathbf{K}}^s) \quad (13)$$

We observed that the optimization converges faster when the data-fitting error is computed in the derivative domain rather than in the intensity domain. This is consistent with the findings of Cho and Lee [27]. On the other hand, the cross-channel prior (see Section IV-A) requires the image to be represented in the intensity domain. Since mixing the intensity domain and the gradient domain in a single subproblem would be too costly, we use the gradient domain only for the kernel subproblem.

The same stochastic random walk algorithm (Algorithm 2) is used for updating the kernel except samples are now drawn from the kernel image. As before, an energy quantum $ed_K$ is added or removed at each sample location $\mathbf{x}_i$ causing the kernel to be updated by Eq. 14, where $\hat{\mathbf{K}}_i^s$ is the estimated kernel at $i^{th}$ iteration of the random walk, and $\delta_{\mathbf{x}_i}$ is the characteristic function (i.e., Kronecker delta function) for the pixel located at $\mathbf{x}_i$. Non-negativity of the kernel is enforced by rejecting any sample that causes a kernel pixel to become negative.

$$\hat{\mathbf{K}}_i^s = \hat{\mathbf{K}}_{i-1}^s \pm ed_K \cdot \delta_{\mathbf{x}_i} \quad (14)$$

Updates to the kernel result in a change to the entire blurry image. To evaluate the data efficiently, the algorithm maintains an estimate of the gradient of the current blurry image, $\nabla_{x,y}\hat{\mathbf{B}}_i^s$, which is compared to the down-sampled captured blurry image to evaluate the change to the objective function. Each sample at $\mathbf{x}_i$ is a scaled Dirac function $\pm ed_K \delta_{\mathbf{x}_i}$, resulting in an update rule whereby a shifted and scaled copy of the current estimate for the intrinsic image $\hat{\mathbf{I}}^s$ is added:

$$\nabla_{x,y}\hat{\mathbf{B}}_i^s = \nabla_{x,y}\hat{\mathbf{B}}_{i-1}^s \pm ed_K(\delta_{\mathbf{x}_i} \otimes \nabla_{x,y}\hat{\mathbf{I}}^s) \quad (15)$$

As in the intrinsic image update, it is necessary to apply a regularizer to the kernel estimation in order to enforce specific properties. For motion blur kernels we expect kernels to be i) smooth, ii) sparse, in the sense that most kernel entries will be zero and iii) continuous, in that the kernel should be a smooth curve over the exposure time. We enforce these properties with the priors from Eq. 16-18 respectively:

- Smoothness [40]:

$$||\nabla^2\hat{\mathbf{K}}^s||_2^2 \quad (16)$$

- Sparsity:

$$||\hat{\mathbf{K}}^s||_p, \quad 0 \le p < 1 \quad (17)$$

- Continuity:

$$||\hat{\mathbf{K}}^s - \mathrm{AD}(\hat{\mathbf{K}}^s)||_2^2 \quad (18)$$

For the continuity prior, Eq. 18, anisotropic diffusion [44] is used for the filter AD(.). This is a non-linear filter that favors long continuous features which helps to reconstruct thin motion-blur trails. A benefit of the stochastic optimization algorithm is that this function may be computed exactly for each sample, rather than linearized per iteration. As with the intrinsic update, the key benefit of the stochastic framework is that only local evaluations of the regularizers are required.

After updating the kernel, a simple denoising filter is applied that sets any pixel in $\hat{\mathbf{K}}^s$ to zero whenever its eight neighboring pixels are near zero. This removes isolated speckles that sometimes occur with the stochastic random walk. The pixels whose intensity is lower than a threshold (i.e., 0.05 times the highest pixel intensity of current estimated kernel) are also set to be zero. This can be interpreted as an additional shrinkage operator that ensures kernel sparsity. The kernel image is normalized to 1 at the end of each iteration.

In Fig. 6, we show empirical convergence test of our method for non-blind kernel estimations. Regarding the parameters in Algorithm 2, we use the initial value of $ed_K$ as 0.01, and $\gamma$ as 0.1 in the experiments.

### C. Updating the Weights

The weights $\theta_{\mathbf{I}}$ and $\theta_{\mathbf{K}}$ define the relative strength of the data-fitting error and regularizers in the objective functions for intrinsic image and kernel updates. The algorithm begins with initially high $\theta_{\mathbf{I}}$ (except for the cross-channel prior explained in Section IV-A) at the coarsest scale and halves them whenever a new scale is started, until minimum thresholds are reached. This helps to avoid local optima in the subproblem. $\theta_{\mathbf{K}}$ is kept unchanged for all scales in our experiments.

After the kernel is estimated at the finest scale, the function $updateIntrinsicImage()$ is applied again to generate the final estimation of intrinsic image $\hat{\mathbf{I}}$ (i.e., line 18, Algo. 1).

In Fig. 7, we visualize the progress of intrinsic and kernel estimation at multi-scale process.

## IV. Algorithmic Extensions

Having described the basic algorithm in Section III we now proceed to introduce several useful extensions, including non-convex priors for color images (Section IV-A) and chromatic kernels (Section IV-B), as well as partially saturated pixels (Section IV-C). Finally, we demonstrate how non-linear versions of the image formation model can also be included to account for non-Gaussian noise models (Section IV-D).

### A. Color Images

To recover color images corrupted by motion blur, a simple extension of the basic algorithm might perform kernel estimation as described in Section III-B (summing the data term over all three channels) followed by separately deblurring each intrinsic channel. However, better results can be obtained by jointly deblurring all intrinsic channels simultaneously since the majority of edges in the true intrinsic image occur in all channels, with sparse hue changes. Based on this observation, Heide *et al.* [46] proposed a cross-channel prior to remove chromatic aberrations caused by low-quality lenses:

$$\sum_{i,j\in\{r,g,b\}} \lambda_{ij}||\hat{\mathbf{I}}_i^s \cdot \nabla_{x,y}\hat{\mathbf{I}}_j^s - \hat{\mathbf{I}}_j^s \cdot \nabla_{x,y}\hat{\mathbf{I}}_i^s||_p, \quad 0 < p \le 1 \quad (19)$$

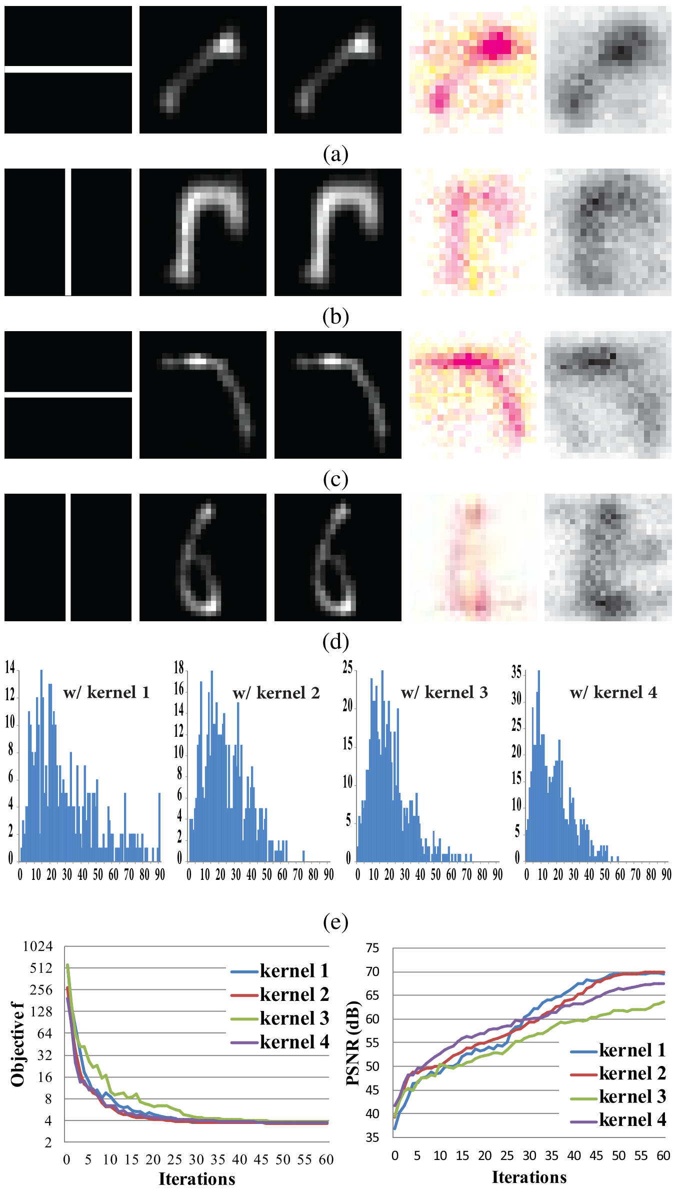


Fig. 6. Example of *non-blind* kernel estimation. The true intrinsic image and kernel are given in Fig. 2. The input blurry images are given in Fig. 3. (a)-(d), from left to right, show the initial kernel, our non-blind estimated kernel, ground truth kernel, distribution of *accepted* sample energy $ed_K$ (values are 10× magnified, *magenta* channel indicates *positive* values and *yellow negative*), and normalized distribution of proposed sample locations $\mathbf{x}_i$ (coded in *key* channel). (e) shows the histogram of the number of proposed samples (including both *accepted* and *rejected* samples). The bottom row show the change of objective function $f$ and PSNR as the number of iterations increases. The initial kernels can be arbitrary for non-blind kernel estimation. Note that the sample energy is non-zero at the region where the pixel intensity is zero in both initial and final recovered kernel. This is due to the post-processing (remove isolated pixel, shrinkage, and normalization) at the end of each iteration. This post-processing also causes non-monotonicity in the objective and PSNR curve. The priors defined in Eq. 16, 17 and 18 were applied. In each iteration, 200 samples were proposed. (a) With kernel 1. (b) With kernel 2. (c) With kernel 3. (d) With kernel 4.

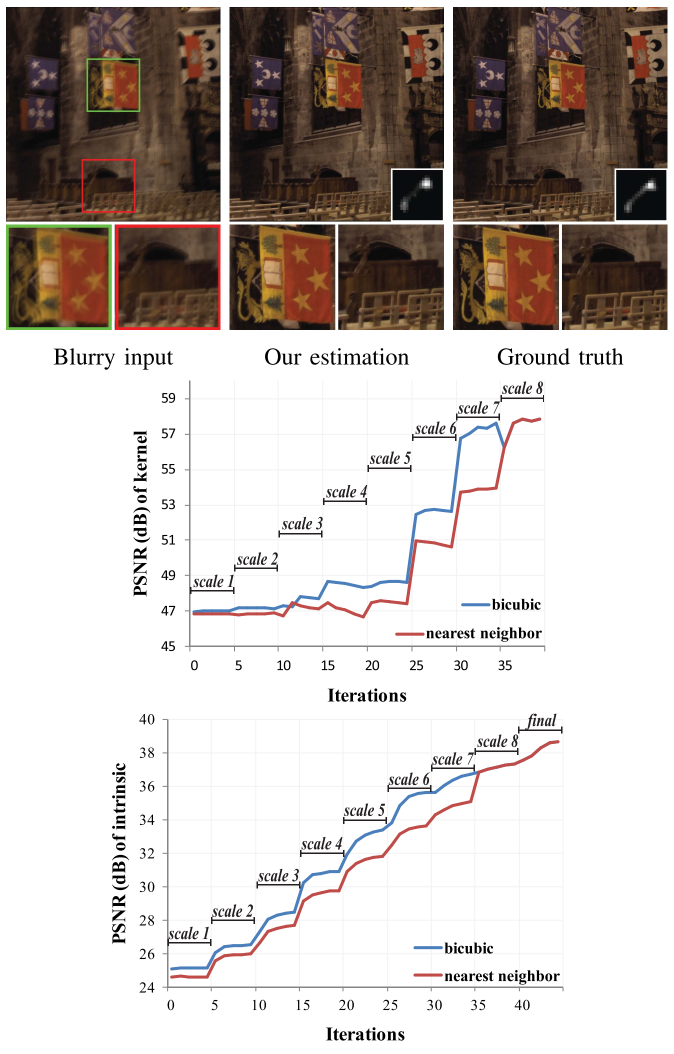


Fig. 7. Example of our blind estimation of intrinsic and kernel. The two plots show the PSNR value of intermediate estimation of intrinsic and kernel at multi-scale scheme. To compute the PSNR values, at each scale we upsample the intrinsic and kernel to the finest resolution by bicubic or nearest neighbor and remove the possible shifts first. The '*final*' step in the plot means the final restoration of the intrinsic image, i.e., line 18, Algorithm 1.

Adding the cross-channel priors to the regularizers $g(.)$ for the intrinsic image results in a non-convex objective. Heide et al. used an alternating minimization in which one channel is deblurred with the other two fixed. We instead reconstruct all channels simultaneously by running one sampling chain per channel run in lock-step. Although the method still alternates between the channels, this occurs so frequently that the optimization is effectively performed simultaneously over all channels. Our algorithm begins with low weight for cross-channel prior at the coarsest scale and doubles it when a new scale is started.

We find that the prior proposed by Heide et al. improves deblurring performance even for achromatic kernels by suppressing color artifacts that would be introduced by separate deblurring of each channel. Example comparisons are shown in Fig. 8, 9 and 10.

## B. Chromatic Kernels

It is further possible to extend the method to estimating chromatic kernels that occur in sensor fusion where individual channels have unique exposure times (see [4]). This is accomplished by separately updating each kernel

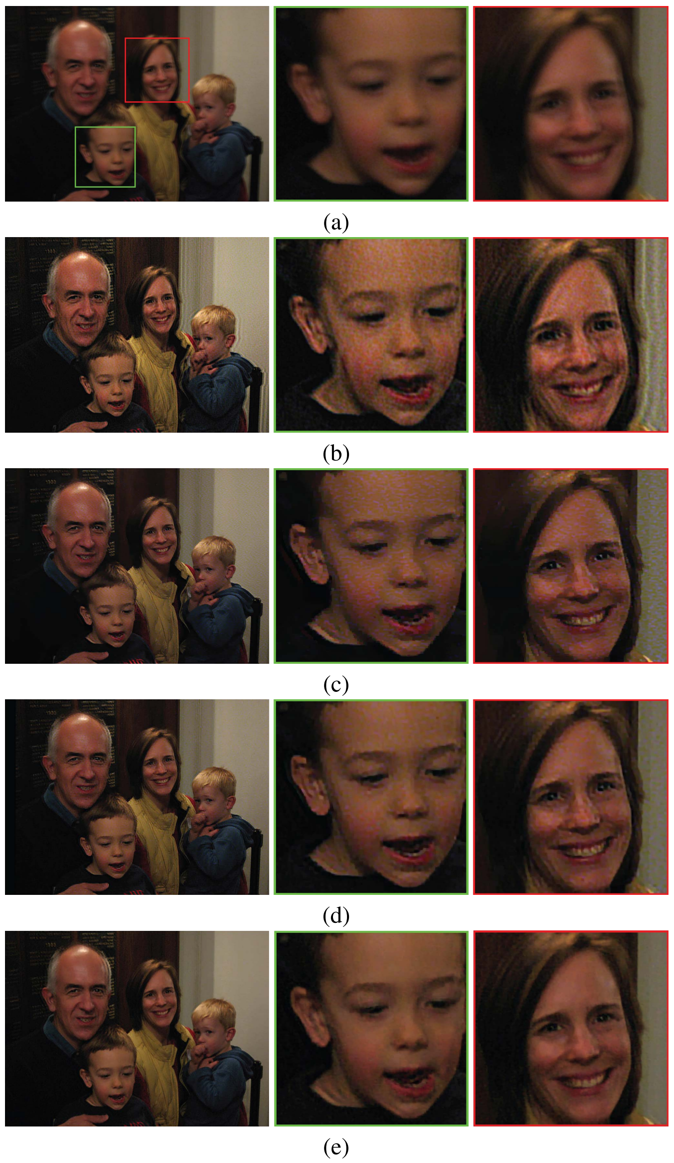


Fig. 8. Results on a noisy real-world image. (a) Input. (b) Fergus *et al.* [13]. (c) Cho and Lee [27]. (d) RDS [45]. (e) Ours. Our algorithm significantly reduces chromatic artifacts compared with previous methods (better view on screen).

channel, but using the cross-channel prior in Eq. 19 during the intrinsic image update at each scale as described in Section IV-A. Synthetic examples of this strategy are shown in Fig. 11.

### *C. Saturated or Missing Data*

Saturated pixels are a common occurrence when taking photos with consumer cameras, as is missing or unreliable data due to lens debris. Deblurring data with saturated pixels often results in visually objectionable ringing artifacts since the capture process clamps the input data in a way that is not consistent with the image formation model, while for debris it may be preferable to mask out such regions and allow the deblurring algorithm to inpaint plausible content. In the following discussion, we consider only the case of saturated blurry pixels, however the approach applies equally well to lens debris.

To handle such saturated pixels, our algorithm performs kernel estimation as usual using all non-saturated pixels. When reconstructing the final intrinsic image, previous work, including [3], simply uses a data term that omits saturated blurry image pixels, leading to improved results over deblurring naively. However we have found that a two-phase approach to intrinsic image estimation yields much improved results.

The two phase algorithm divides the intrinsic image into two regions: a *reliable* region which does not contribute to saturated blurry image pixels and an *unreliable* region that contains pixels which contribute to saturated blurry pixels. Four binary masks are defined:

- $\mathbf{M}_\mathbf{s}^\mathbf{B}$ Saturated pixels in the blurred input.
- $\mathbf{M}_\mathbf{v}^\mathbf{I}$ Mask of unreliable intrinsic pixels with a saturated pixel from $\mathbf{M}_\mathbf{s}^\mathbf{B}$ in their support.
- $\mathbf{M}_\mathbf{u}^\mathbf{I}$ Mask of reliable intrinsic pixels, the inverse mask of $\mathbf{M}_\mathbf{v}^\mathbf{I}$.
- $\mathbf{M}_\mathbf{d}^\mathbf{B}$ Mask of blurred pixels with a contribution from an unreliable pixel, i.e. where $\mathbf{K} \otimes \mathbf{M}_\mathbf{v}^\mathbf{I} \neq 0$.

Using these masks we perform the intrinsic image reconstruction in two phases. First the intrinsic image is estimated for reliable intrinsic image pixels in $\mathbf{M}_\mathbf{u}^\mathbf{I}$, masking out data term contributions from unreliable blurred pixels in $\mathbf{M}_\mathbf{d}^\mathbf{B}$ by minimizing:

$$f(\hat{\mathbf{I}}) = ||(\mathbf{B} - \hat{\mathbf{K}} \otimes \hat{\mathbf{I}}) \cdot (\mathbf{1} - \mathbf{M}_\mathbf{d}^\mathbf{B})||_2^2 + \theta_\mathbf{I} \cdot g(\hat{\mathbf{I}}) \qquad (20)$$

This optimization outputs the estimated intrinsic image everywhere that the linear image formation model holds, generating samples everywhere in the image as needed to minimize both data term *and* the priors $g(.)$. The second phase reconstructs the unreliable regions, leaving the intrinsic image in the reliable regions (i.e. in $\mathbf{M}_\mathbf{u}^\mathbf{I}$) fixed.

$$f(\hat{\mathbf{I}}) = ||(\mathbf{B} - clip(\hat{\mathbf{K}} \otimes \hat{\mathbf{I}})) \cdot \mathbf{M}_\mathbf{d}^\mathbf{B}||_2^2 + \theta_\mathbf{I} \cdot g(\hat{\mathbf{I}}) \qquad (21)$$

The second phase only generates samples within the mask $\mathbf{M}_\mathbf{v}^\mathbf{I}$. By performing the reconstruction in this method, ringing is constrained to the non-reliable image region unlike in the typical approach where it can spread well beyond as a consequence of the data fitting term. Fig. 12 shows our results on synthetic partially saturated data. When dealing with color images, the proposed two-phase reconstruction is the same as described except that the masks vary in different color channels.

### *D. Poisson Noise*

In previous sections, we use quadratic fidelity in the objective by assuming white Gaussian noise in the input images. Here we extend our algorithm to deal with images containing Poisson noise, using the Anscome transform [47]:

$$Ansc(\mathbf{z}) = 2\sqrt{\mathbf{z} \cdot c + 3/8}, \qquad (22)$$

where $\mathbf{z}$ is normalized pixel intensity, $c$ is a scalar for converting $\mathbf{z}$ to its corresponding photon number.

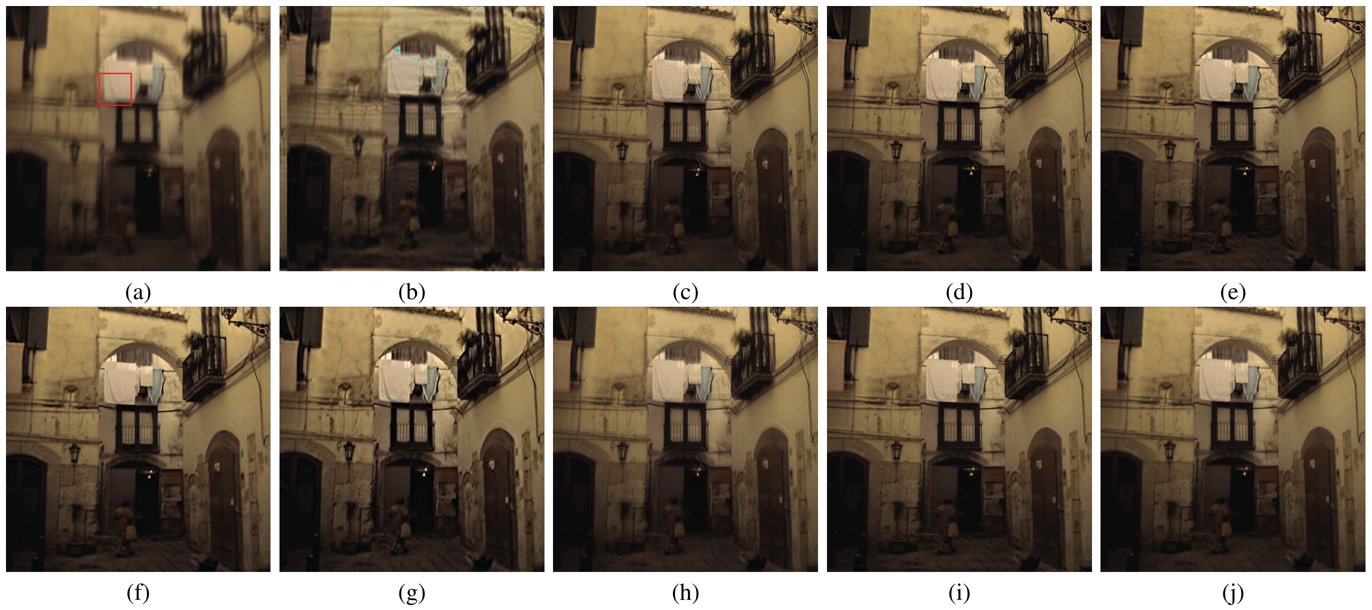


Fig. 9. Results on a real-world image and visual comparisons of state-of-the-art methods. (a) Input. (b) Fergus *et al.* [13]. (c) Shan *et al.* [22]. (d) Cho and Lee [27]. (e) Xu and Jia [30]. (f) Hirsch *et al.* [37]. (g) Krishnan *et al.* [23]. (h) Whyte *et al.* [34]. (i) Xu *et al.* [24]. (j) Ours. A closeup is shown in Fig. 10.

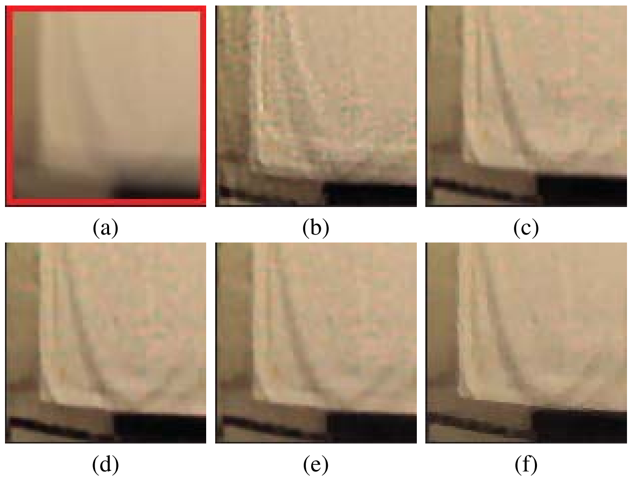


Fig. 10. Closeup of the cloth region in the scene shown in Fig. 9. (a) Input. (b) Shan *et al.* [22]. (c) Cho and Lee [27]. (d) Xu and Jia [30]. (e) Xu *et al.* [24]. (f) Ours. Our result contains much less chromatic artifacts than the other methods.

The Anscombe transform (denoted as Ansc(.)) converts Poisson noise to approximately Gaussian noise. It allows us to use quadratic term for data-fitting error in the transform domain, and thus fits our multi-scale framework. Similar transformations are available for other noise models, including mixtures of Gaussian and Poisson noise, which are common in real images.

Specifically, as shown in Eq. 23 and 24, we apply the Anscombe transform on the observed blurry image $\mathbf{B}$ before downsampling it into each scale $s$. At each scale, the data-fitting error is computed in the transform domain, while the regularizers on the intrinsic image $g(\hat{\mathbf{I}}^s)$ are still computed in regular domain. This is because all the intrinsic image priors were learned from natural images, and may not hold well in the transform domain.

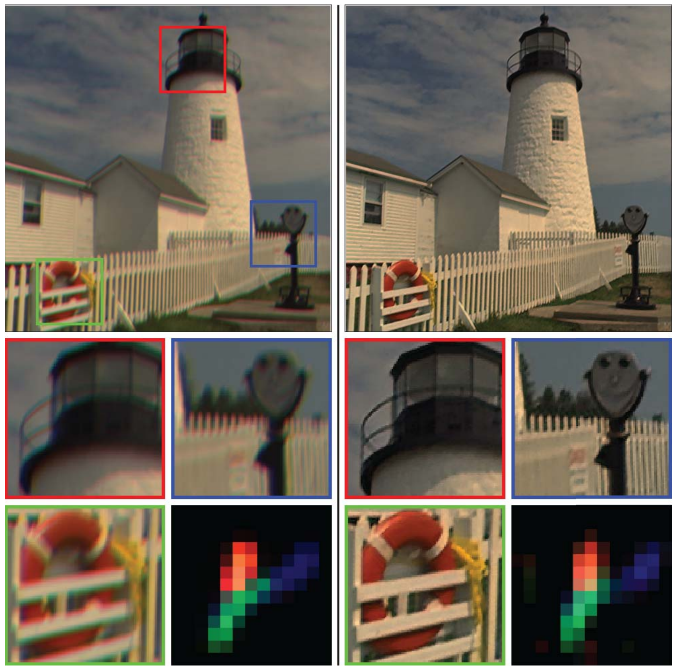
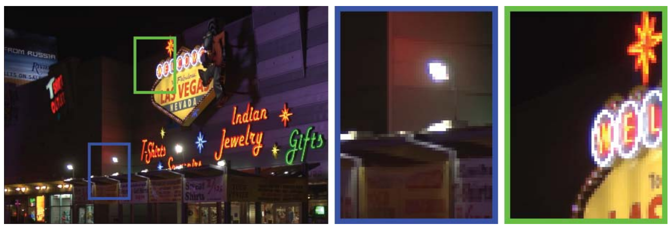


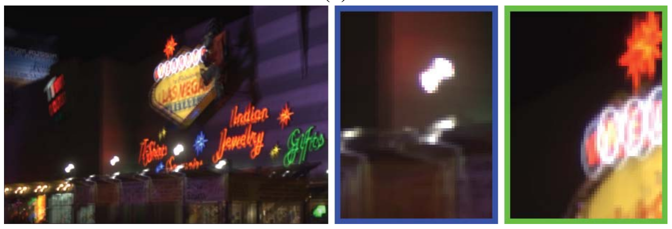
Fig. 11. Results on chromatic blur kernel. *Left column*: input image blurred with a chromatic kernel (right bottom); *Right column*: our recovered intrinsic image and blur kernel (right bottom).

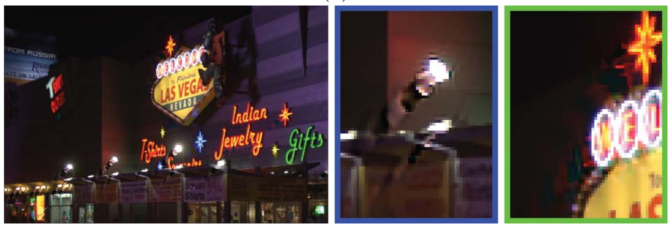
$$f(\hat{\mathbf{I}}^s) = ||(Ansc(\mathbf{B}))^s - Ansc(\hat{\mathbf{K}}^s \otimes \hat{\mathbf{I}}^s))||_2^2 + \theta_{\mathbf{I}} \cdot g(\hat{\mathbf{I}}^s) \quad (23)$$

$$f(\hat{\mathbf{K}}^s) = ||\nabla_{x,y}(Ansc(\mathbf{B}))^s - \nabla_{x,y}Ansc(\hat{\mathbf{K}}^s \otimes \hat{\mathbf{I}}^s)||_2^2 + \theta_{\mathbf{K}} \cdot g(\hat{\mathbf{K}}^s) \quad (24)$$

Note that the data-fitting terms are non-convex now. In Fig. 13, we show a synthetic example with Poisson noise


(a)


(b)


(c)

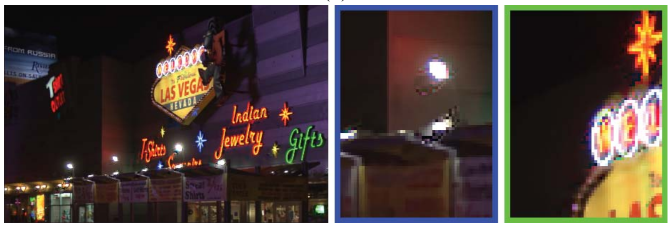

(d)

Fig. 12. Results on partially saturated image. (a) True intrinsic image. (b) Input blurry image. (c) Recovered intrinsic without two-phase approach. (d) Recovered intrinsic with two-phase approach. The dynamic range of pixel intensities in the true intrinsic image is [0, 68]. The simulated blurry image is clamped to 1. The proposed two-phase approach helps reduce ringing artifacts near saturated pixels.

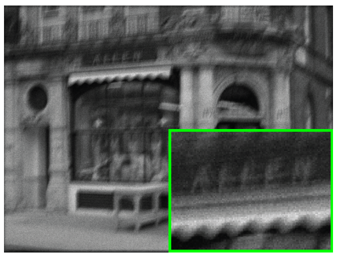
(peak intensity = 500). We run our framework with Gaussian noise assumption (using Eq. 2, 13), and with Poisson noise assumption (using Eq. 23, 24). The latter produces visually and quantitatively better results.

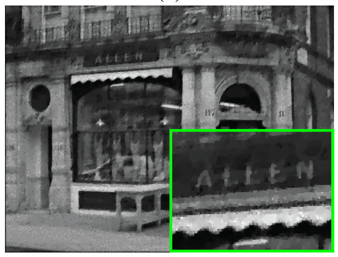
## V. Results

### A. Visual Comparisons

In Fig. 8, 9, 10, and 16 we compare the results of our algorithm with several state-of-the-art methods on real-world images. Our algorithm significantly reduces chromatic artifacts in the recovered intrinsic images. Fig. 11 shows our results on simulated data with chromatic blur. Our algorithm recovers the chromatic kernel well and produces a clean intrinsic image without color artifacts. Fig. 12 contains results for partially saturated data. The proposed simple two-phase method effectively reduces the ringing artifacts near the saturated pixels. Fig. 13 shows our results with the proposed non-convex data-fitting error in Eq. 23 and 24.

(a)


(b)


(c)

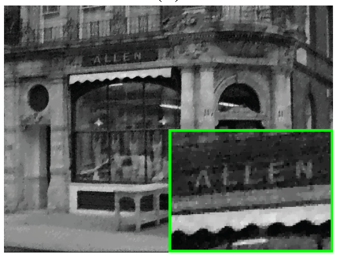

(d)

Fig. 13. Results on a synthetic image with Poisson noise (peak intensity = 500). Each subfigure contains an inset at the right-bottom corner for better view. (a) True intrinsic. (b) Input blurry (21.23 dB). (c) Our result with Gaussian noise assumption (24.24 dB). (d) Our result with Poisson noise assumption and the Anscombe transform (24.70 dB). The result with Poisson noise assumption recovers more details and is less noisy especially in bright regions. We run both experiments with numerious parameters and select the results with highest PSNR.

### B. Qualitative Comparisons

We also tested our algorithm on a real-world database that contains ground truth intrinsic images [48]. The database consists of 4 images, each of which is blurred with 12 kernels. 8 of the kernels are approximately uniform across the image, while 4 kernels are both large and exhibit strong spatial variation. The dataset provides 199 unblurred frames recorded during the camera motion trajectory for each image. We run the provided script to compute the Peak Signal-to-Noise Ratio (PSNR) value for each of our results. The script first estimate the optimal intensity scaling and translation between the recovered image and the unblurred reference image such that their $l_2$ error over three color channels becomes minimal. PSNR is then computed using these calibrated images. The final PSNR values reported in the paper is defined as the maximum PSNR between the recovered image and any of the 199 unblurred reference images along the trajectory.

In Table I, we show the PSNR values averaged over all 4 images for each kernel by our algorithm, and compare with Fergus *et al.* [13], Shan *et al.* [22], Cho and Lee [27], Xu and Jia [30], Krishnan *et al.* [23], Whyte *et al.* [34], Hirsch *et al.* [37] and the Robust Deblurring Software (RDS) [45]. The downloadable software [45] incorporates technologies proposed in [24] and [30]. We adjusted the parameters in their software to produce the best results possible. For the other methods we use the published PSNR results directly from the database [48]. The PSNR value and recovered intrinsic image for each image can be found in the supplementary material.

On mostly spatially-invariant kernels (#1-7 and #12), our algorithm produces results that are either close to or

TABLE I

PSNR (dB) COMPARISONS ON THE BENCHMARK DATASET [48]: ON MOSTLY SPATIALLY-INVARIANT KERNELS (#1-7 AND #12), OUR ALGORITHM PRODUCES RESULTS CLOSE TO OR BETTER THAN THE STATE-OF-THE-ART METHODS. OUR ALGORITHM FAILS TO PRODUCE THE BEST RESULTS ON THE EXTREMELY LARGE AND SPATIALLY-VARIANT KERNELS #8-11 (WITH SIZE OVER 141 BY 141 PIXELS). THE RESOLUTION OF EACH INPUT IMAGE IS 800 BY 800 PIXELS. PLEASE SEE SECTION V FOR MORE DETAILS

| | Kernel 01 | Kernel 02 | Kernel 03 | Kernel 04 |
|---|---|---|---|---|
| **Kernel width** | 35 | 35 | 17 | 35 |
| Input | 24.89 | 27.85 | 32.34 | 28.09 |
| Fergus [13] | 20.63 | 29.38 | 31.51 | 29.48 |
| Shan [22] | 29.42 | 28.28 | 30.77 | 28.60 |
| Cho [27] | 32.49 | 31.86 | 31.44 | 30.89 |
| Xu and Jia [30] | 32.45 | 32.58 | 32.22 | 32.01 |
| Krishnan [23] | 31.57 | 31.09 | 31.67 | 30.77 |
| Whyte [34] | 32.08 | 32.20 | 34.61 | 32.10 |
| Hirsch [37] | 32.04 | 30.09 | 33.94 | 32.13 |
| RDS [45] | 31.94 | 31.71 | 31.42 | 31.30 |
| Ours | **32.65** | **32.68** | **35.35** | **33.74** |
| | Kernel 05 | Kernel 06 | Kernel 07 | Kernel 12 |
| **Kernel width** | 35 | 35 | 35 | 49 |
| Input | 29.22 | 25.22 | 24.94 | 23.85 |
| Fergus [13] | 25.76 | 22.34 | 20.70 | 20.11 |
| Shan [22] | 30.04 | 26.85 | 26.83 | 23.76 |
| Cho [27] | 32.38 | 30.01 | 30.91 | 28.98 |
| Xu and Jia [30] | **32.98** | 30.43 | **31.40** | 29.51 |
| Krishnan [23] | 30.58 | 24.59 | 25.80 | 23.32 |
| Whyte [34] | 32.54 | 30.89 | 29.06 | 28.21 |
| Hirsch [37] | 32.82 | 29.53 | 29.32 | 26.81 |
| RDS [45] | 31.78 | 30.06 | 30.87 | 29.18 |
| Ours | 32.56 | **31.74** | 30.96 | **30.12** |
| | Kernel 08 | Kernel 09 | Kernel 10 | Kernel 11 |
| **Kernel width** | 141 | 141 | 141 | 141 |
| Input | 19.55 | 19.82 | 20.50 | 22.90 |
| Fergus [13] | 17.93 | 18.87 | 18.72 | 16.95 |
| Shan [22] | 19.19 | 22.90 | 20.49 | 23.56 |
| Cho [27] | 22.34 | 28.00 | 23.96 | 24.54 |
| Xu and Jia [30] | **22.54** | **28.35** | **24.12** | **25.87** |
| Krishnan [23] | 15.35 | 22.14 | 20.15 | 21.68 |
| Whyte [34] | 19.07 | 19.80 | 20.38 | 23.19 |
| Hirsch [37] | 19.49 | 23.50 | 20.19 | 23.40 |
| RDS [45] | 19.52 | 26.46 | 21.77 | 25.77 |
| Ours | 19.79 | 21.16 | 21.02 | 22.88 |

better than the best published methods. We notice that the state-of-the-art RDS [45]'s results sometimes look sharper than ours, but are actually oversharpened at the edges and thus have lower PSNRs. We show examples in Fig. 14, where

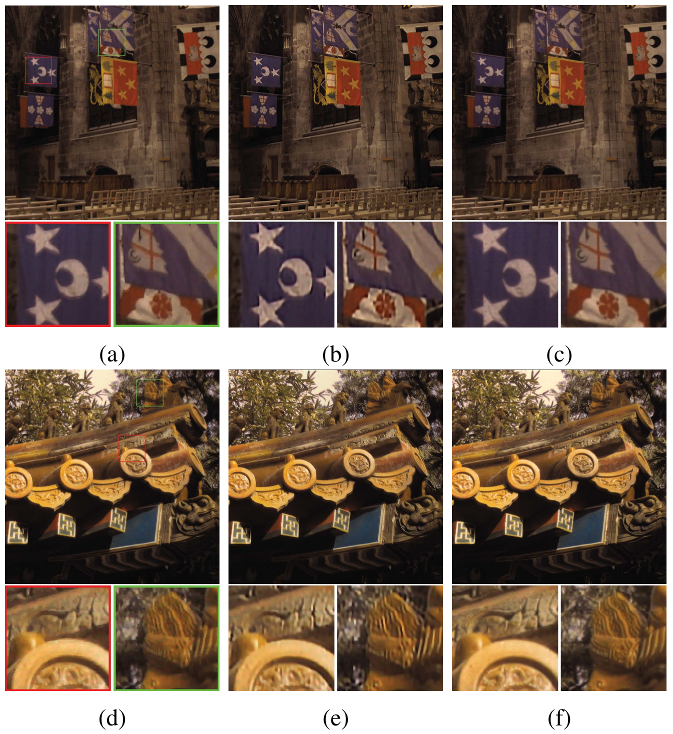


Fig. 14. Comparison on image #1 with kernel #3 and image #4 with kernel #4 from the dataset [48]. (a) Ours (38.82dB). (b) RDS [45] (33.34dB). (c) Reference. (d) Ours (33.29dB) (e) RDS [45] (30.72dB). (f) Reference. The filter-based method RDS [45] over-sharpens the image and creates halo artifacts near the edges (better view on screen).

their results even look sharper than the ground truth and contain halo artifacts at the edge pixels. This may be caused by the use of explicit shock filter and bilateral filter in their kernel estimation.

All methods perform significantly worse on the spatially-variant kernels #8-11. Our software does not currently deal with spatially-variant kernels, and instead recovers an average kernel for the whole image. As a result, our method only produces PSNR values in the middle of the field for these kernels. Fixing this problem would require cutting the image into tiles, solving a blind deconvolution problem for each tile, realigning the resulting tiles (since blind deconvolution can introduce an offset in the kernel and intrinsic image), and stitching the results back together. This should be easily feasible in the future, but is not currently implemented.

## C. Computational Cost

We compared the runtime with two state-of-the-art methods, Cho and Lee [27] and RDS [45], using the executable files provided by the authors. The method by Cho and Lee [27] requires only a few seconds per megapixel, but their results are consistently worse than ours (see Fig. 8, 9 and 10 and Table I). In Fig. 8 for an 848 by 636 blurry/noisy RGB image and a 19 by 19 achromatic kernel, our method requires 197.0 seconds in total (140.4 seconds for blind kernel estimation, and 56.6 seconds for non-blind deconvolution). RDS method [45] is relatively faster than ours at 121.9 seconds, but their results show suffer from more artifacts.

TABLE II

Running Time Analysis: The Reported Time Are in Seconds. We Run Our Unoptimized Code on Images With Pixel Resolution $400 \times 400$, $800 \times 800$, $1200 \times 1200$, and Kernels With Pixel Resolution $15 \times 15$, $25 \times 25$, $35 \times 35$. The Blurry Images Are Simulated With Resized Intrinsic Image and Blur Kernel at Each Resolution. (a) Shows the Experiments on Gray-Scaled Images. (b) Shows the Experiments on RGB Images, Where All Channels Were Recovered Simultaneously. We Increased the Number of Proposed Samples as the Image Size and Kernel Size Increase. The Result Images and Parameters Are Shown in the Supplementary

(a)

| Image width | Kernel width | Multiscale estimation | | Final restoration | Total |
|---|---|---|---|---|---|
| | | Intrinsic update | Kernel update | | |
| 400 | 15 | 4.68 | 7.61 | 2.92 | 15.71 |
| | 25 | 13.44 | 14.58 | 5.45 | 34.03 |
| | 35 | 23.24 | 21.44 | 15.74 | 61.12 |
| 800 | 15 | 11.53 | 54.64 | 7.05 | 74.93 |
| | 25 | 19.94 | 59.64 | 14.24 | 95.48 |
| | 35 | 53.81 | 80.31 | 37.03 | 172.99 |
| 1200 | 15 | 18.15 | 132.56 | 13.06 | 167.85 |
| | 25 | 30.63 | 139.03 | 25.81 | 199.09 |
| | 35 | 75.55 | 186.19 | 53.85 | 319.42 |

(b)

| Image width | Kernel width | Multiscale estimation | | Final restoration | Total |
|---|---|---|---|---|---|
| | | Intrinsic update | Kernel update | | |
| 400 | 15 | 13.02 | 25.96 | 8.13 | 47.62 |
| | 25 | 36.67 | 49.74 | 16.24 | 103.21 |
| | 35 | 65.02 | 72.77 | 47.33 | 185.81 |
| 800 | 15 | 32.58 | 180.57 | 19.94 | 234.81 |
| | 25 | 56.32 | 187.81 | 40.33 | 286.04 |
| | 35 | 155.47 | 439.09 | 116.72 | 713.16 |
| 1200 | 15 | 48.99 | 402.35 | 35.83 | 491.09 |
| | 25 | 83.42 | 408.32 | 71.85 | 567.12 |
| | 35 | 217.86 | 594.26 | 154.54 | 971.14 |

We also run our algorithm on different size images and kernels and report the runtime in Table II. The code was compiled with gcc and the experiments were done on an Intel i7 CPU with 16GB RAM. The result images and parameters are shown in the supplementary document.

### D. Influence of the Noise

To test the influence of noise on the performance, we run our algorithm on synthetic data with various noise levels. The parameters are tuned roughly for the results. The results are shown in Fig. 15.

### E. Priors and Parameter Selection

Our framework allows us to easily adapt any priors or data-fitting term in the objective function. We use

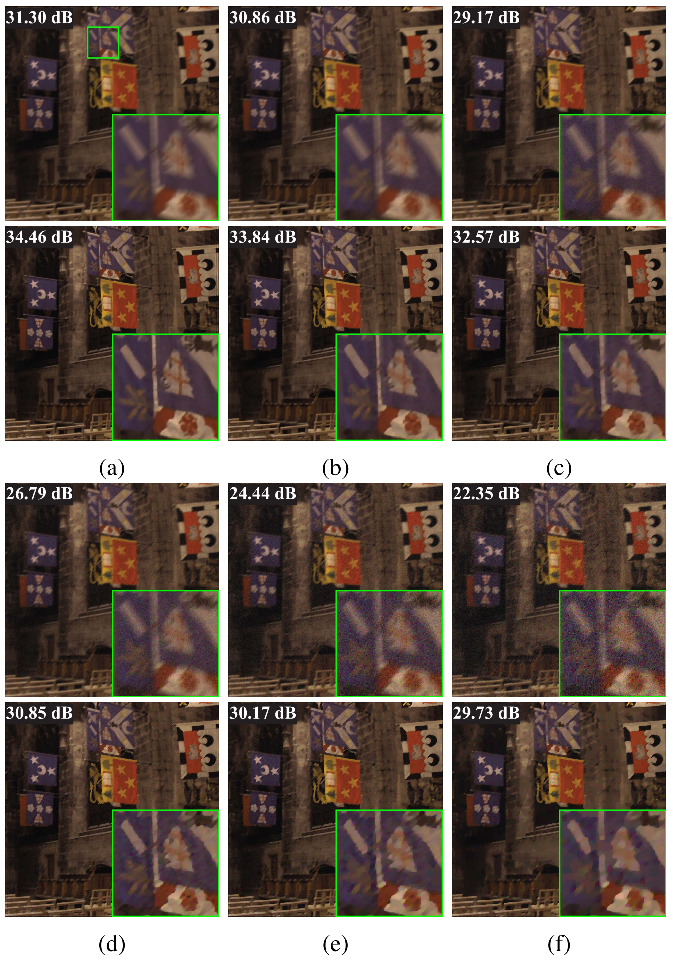


Fig. 15. Results on synthetic data with difference levels of white Gaussian noise. In subfigure (a-f), the standard derivation ($\sigma$) of the noise is set to be 0, 0.01, 0.02, 0.03, 0.04 and 0.05 respectively. In each subfigure, the 1st row shows the input blurry image, and the 2nd row shows our deblurred result. The PSNR values are shown at the left-top corner of each image. An inset is shown at the right-bottom corner of each image for better view.

*smoothness* (Eq. 16), *sparsity* (Eq. 17, p = 0.8) and *continuity* (Eq. 18) as kernel priors for all results in the paper. We use *sparse gradient* (Eq. 11, p = 0.6 or 0.8), *sparse second-order derivatives* (Eq. 12, p = 1) and *cross-channel prior* (Eq. 19, p = 1) as intrinsic image priors for Fig. 7, 8, 9, 10 and Table I. And we use *isotropic total variation* (Eq. 9) and *cross-channel prior* (Eq. 19, p = 1) as intrinsic image priors for Fig. 11.

Regarding the number of iterations and sample mutations in our experiments, we usually use $T$ (in Algorithm 1) as 5-10, $N$ (in Algorithm 2) as 5 for both intrinsic and kernel updating, and $M$ (in Algorithm 2) as 10000-50000 for intrinsic updating and 100-500 for kernel updating in the multi-scale process. In the final intrinsic image restoration step (line 18, Algo. 1), we usually use $N$ as 5, and $M$ as 100000-500000. These numbers are proportionally adjusted with the resolution of input image and kernel for better convergence.

The prior weights $\theta_{\mathbf{I}}$ and $\theta_{\mathbf{K}}$ are roughly tuned for best results. In our experiments we set the initial and minimum-threshold weight of *sparse gradient* as 0.05-0.5 and 0.0005-0.002, the initial and maximum-threshold weight of

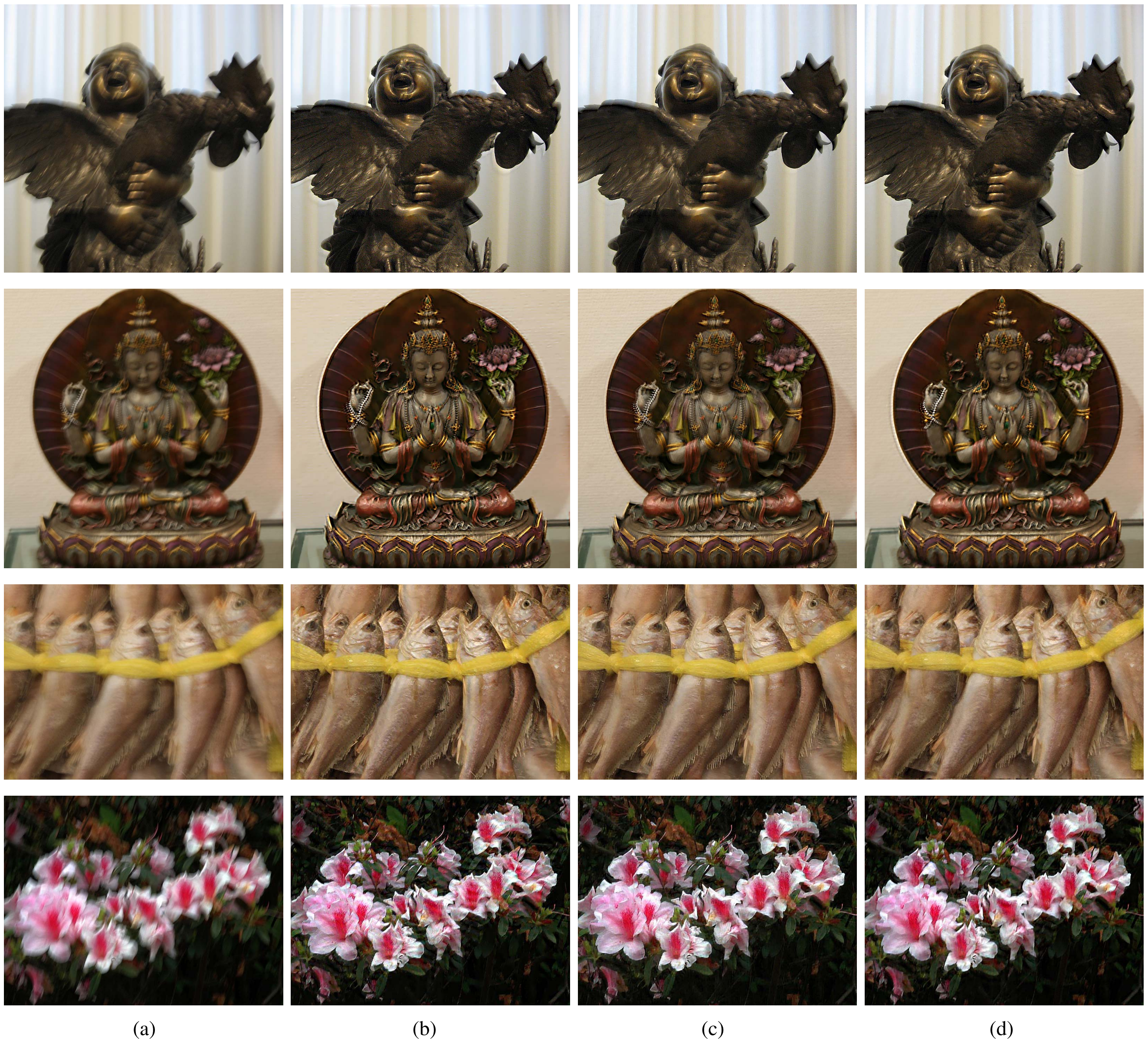


Fig. 16. Results on more real data from [27] and [45] (better view on screen). (a) Blurry input. (b) Cho and Lee [27]. (c) RDS [45]. (d) Ours.

*cross-channel prior* as 0.0001 and 0.0005 (see Section III-C for the strategy of weights updating). In the final intrinsic image restoration step, we set the weight of *sparse gradient* and *cross-channel prior* as 0.0005-0.002 and 0.0005-0.002.

## VI. Conclusion and Future Work

In this paper we present an attempt using simple random search technique for complex imaging problems: a simple and effective algorithm for blind motion deblurring from a single input image. We propose to use cross-channel information to reduce chromatic artifacts in the estimated intrinsic images and to recover chromatic blur kernels. We also propose a two-phase method to reduce ringing artifacts when deblurring saturated or missing pixels. Furthermore, we propose to use a non-convex data-fitting term to deal with Poisson noisy images.

Our algorithm provides an easy-to-use framework for blind deconvolution problems. It allows us to easily test new priors for both the kernel and the intrinsic image. This kind of experimentation would be much harder with other optimization methods.

The computational efficiency of our method is below those highly optimized specialized solvers. However, such solvers typically include only a single regularization term, whereas we can easily combine many, for much improved image quality.

In the future, we would like to extend our algorithm to handle spatially variant kernels with the method outlined above. We also would like to further improve on the handling of saturated pixels. We observe that pixels saturated in one channel might not be saturated in another. By employing the cross-channel information together with neighboring pixels, we should be able to recover the saturated pixels better.

## Acknowledgment

The authors would like to thank the associate editor and anonymous reviewers for their valuable comments.

## References

[1] N. Joshi, S. B. Kang, C. L. Zitnick, and R. Szeliski, "Image deblurring using inertial measurement sensors," *ACM Trans. Graph.*, vol. 29, no. 4, p. 30, Jul. 2010.

[2] A. Rav-Acha and S. Peleg, "Two motion-blurred images are better than one," *Pattern Recognit. Lett.*, vol. 26, no. 3, pp. 311–317, Feb. 2005.

[3] J. Gregson, F. Heide, M. B. Hullin, M. Rouf, and W. Heidrich, "Stochastic deconvolution," in *Proc. IEEE Conf. CVPR*, Jun. 2013, pp. 1043–1050.

[4] L. Lelégard, E. Delaygue, M. Brédif, and B. Vallet, "Detecting and correcting motion blur from images shot with channel-dependent exposure time," in *Proc. 22nd ISPRS Congr.*, vol. 1, no. 3, pp. 341–346, 2012.

[5] R. Raskar, A. Agrawal, and J. Tumblin, "Coded exposure photography: Motion deblurring using fluttered shutter," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 795–804, Jul. 2006.

[6] A. Levin, R. Fergus, F. Durand, and W. T. Freeman, "Image and depth from a conventional camera with a coded aperture," *ACM Trans. Graph.*, vol. 26, no. 3, p. 70, Jul. 2007.

[7] S. K. Nayar and M. Ben-Ezra, "Motion-based motion deblurring," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 6, pp. 689–698, Jun. 2004.

[8] A. Moretti *et al.*, "In-flight calibration of the Swift XRT point spread function," *Proc. SPIE*, vol. 5898, pp. 348–356, Jan. 2005.

[9] J. W. Shaevitz and D. A. Fletcher, "Enhanced three-dimensional deconvolution microscopy using a measured depth-varying point-spread function," *J. Opt. Soc. Amer. A, Opt. Image Sci.*, vol. 24, no. 9, pp. 2622–2627, 2007.

[10] A. Chambolle and T. Pock, "A first-order primal-dual algorithm for convex problems with applications to imaging," *J. Math. Imag. Vis.*, vol. 40, no. 1, pp. 120–145, May 2011.

[11] I. Daubechies, M. Defrise, and C. D. Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Commun. Pure Appl. Math.*, vol. 57, no. 11, pp. 1413–1457, Nov. 2004.

[12] T. Goldstein and S. Osher, "The split Bregman method for L1-regularized problems," *SIAM J. Imag. Sci.*, vol. 2, no. 2, pp. 323–343, 2009.

[13] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman, "Removing camera shake from a single photograph," in *Proc. SIGGRAPH*, 2006, pp. 787–794.

[14] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *Proc. IEEE Conf. CVPR*, vol. 2. Jun. 2005, pp. 60–65.

[15] K. Dabov, A. Foi, V. Katkovnik, K. Egiazarian, and K. Egiazarian, "BM3D image denoising with shape-adaptive principal component analysis," in *Proc. SPARS*, Apr. 2009.

[16] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *Proc. ICCV*, Sep./Oct. 2009, pp. 2272–2279.

[17] A. Danielyan, V. Katkovnik, and K. Egiazarian, "Image deblurring by augmented Lagrangian with BM3D frame prior," in *Proc. WITMSE*, Aug. 2010, pp. 16–18.

[18] S. Kindermann, S. Osher, and P. W. Jones, "Deblurring and denoising of images by nonlocal functionals," *Multiscale Model. Simul.*, vol. 4, no. 4, pp. 1091–1115, 2005.

[19] J. Miskin and D. J. MacKay, "Ensemble learning for blind image separation and deconvolution," in *Advances In Independent Component Analysis*. Berlin, Germany: Springer-Verlag, 2000, pp. 123–141.

[20] W. H. Richardson, "Bayesian-based iterative method of image restoration," *J. Opt. Soc. Amer.*, vol. 62, no. 1, pp. 55–59, 1972.

[21] L. B. Lucy, "An iterative technique for the rectification of observed distributions," *Astronomical J.*, vol. 79, no. 2, pp. 745–754, 1974.

[22] Q. Shan, J. Jia, and A. Agarwala, "High-quality motion deblurring from a single image," in *Proc. SIGGRAPH*, 2008, pp. 73:1–73:10.

[23] D. Krishnan, T. Tay, and R. Fergus, "Blind deconvolution using a normalized sparsity measure," in *Proc. IEEE Conf. CVPR*, Jun. 2011, pp. 233–240.

[24] L. Xu, S. Zheng, and J. Jia, "Unnatural $L_0$ sparse representation for natural image deblurring," in *Proc. IEEE Conf. CVPR*, Jun. 2013, pp. 1107–1114.

[25] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 123–231, Jan. 2014.

[26] J. Pan, Z. Hu, Z. Su, and M.-H. Yang, "Deblurring text images via $L_0$-regularized intensity and gradient prior," in *Proc. IEEE Conf. CVPR*, Jun. 2014, pp. 2901–2908.

[27] S. Cho and S. Lee, "Fast motion deblurring," in *Proc. SIGGRAPH Asia*, 2009, pp. 145:1–145:8.

[28] S. Osher and L. I. Rudin, "Feature-oriented image enhancement using shock filters," *SIAM J. Numer. Anal.*, vol. 27, no. 4, pp. 919–940, 1990.

[29] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. ICCV*, Jan. 1998, pp. 839–846.

[30] L. Xu and J. Jia, "Two-phase kernel estimation for robust motion deblurring," in *Proc. ECCV*, 2010, pp. 157–170.

[31] Y. Wang and W. Yin, "Sparse signal reconstruction via iterative support detection," *SIAM J. Imag. Sci.*, vol. 3, no. 3, pp. 462–491, 2010.

[32] Y.-W. Tai and S. Lin, "Motion-aware noise filtering for deblurring of noisy and blurry images," in *Proc. IEEE Conf. CVPR*, Jun. 2012, pp. 17–24.

[33] L. Zhong, S. Cho, D. Metaxas, S. Paris, and J. Wang, "Handling noise in single image deblurring using directional filters," in *Proc. IEEE Conf. CVPR*, Jun. 2013, pp. 612–619.

[34] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce, "Non-uniform deblurring for shaken images," in *Proc. IEEE Conf. CVPR*, Jun. 2010, pp. 491–498.

[35] A. Gupta, N. Joshi, C. L. Zitnick, M. Cohen, and B. Curless, "Single image deblurring using motion density functions," in *Proc. ECCV*, 2010, pp. 171–184.

[36] S. Harmeling, H. Michael, and B. Schölkopf, "Space-variant single-image blind deconvolution for removing camera shake," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 829–837.

[37] M. Hirsch, C. J. Schuler, S. Harmeling, and B. Scholkopf, "Fast removal of non-uniform camera shake," in *Proc. ICCV*, Nov. 2011, pp. 463–470.

[38] J. Gregson, M. Krimerman, M. B. Hullin, and W. Heidrich, "Stochastic tomography and its applications in 3D imaging of mixing fluids," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 52:1–52:10, 2012.

[39] G. Casella and C. P. Robert, *Monte Carlo Statistical Methods*. New York, NY, USA: Springer-Verlag, 1999.

[40] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman, "Understanding and evaluating blind deconvolution algorithms," in *Proc. IEEE Conf. CVPR*, Jun. 2009, pp. 1964–1971.

[41] S. Shalev-Shwartz and A. Tewari, "Stochastic methods for $\ell_1$-regularized loss minimization," *J. Mach. Learn. Res.*, vol. 12, pp. 1865–1892, Feb. 2011.

[42] S. Sardy, A. G. Bruce, and P. Tseng, "Block coordinate relaxation methods for nonparametric wavelet denoising," *J. Comput. Graph. Statist.*, vol. 9, no. 2, pp. 361–379, 2000.

[43] Y. Li and S. Osher, "Coordinate descent optimization for $\ell_1$ minimization with application to compressed sensing; a greedy algorithm," *Inverse Problems Imag.*, vol. 3, no. 3, pp. 487–503, 2009.

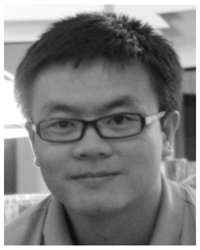
[44] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 7, pp. 629–639, Jul. 1990.

[45] *Robust Deblurring Software*. [Online]. Available: http://www.cse.cuhk.edu.hk/~leojia/deblurring.htm, accessed Jan. 1, 2014.

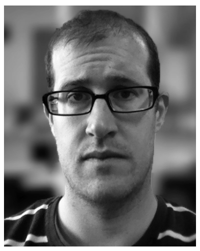
[46] F. Heide, M. Rouf, M. B. Hullin, B. Labitzke, W. Heidrich, and A. Kolb, "High-quality computational imaging through simple lenses," *ACM Trans. Graph.*, vol. 32, no. 5, p. 149, Sep. 2013.

[47] F. Anscombe, "The transformation of Poisson, binomial and negative-binomial data," *Biometrika*, vol. 35, nos. 3–4, pp. 246–254, Dec. 1948.

[48] R. Köhler, M. Hirsch, B. Mohler, B. Schölkopf, and S. Harmeling, "Recording and playback of camera shake: Benchmarking blind deconvolution with a real-world database," in *Proc. ECCV*, 2012, pp. 27–40.



**Lei Xiao** received the B.S. degree in biomedical engineering from the Huazhong University of Science and Technology, in 2009, and the M.S. degree in computer engineering from the University of New Mexico, in 2012. He is currently pursuing the Ph.D. degree with the Computer Science Department, University of British Columbia. His research focuses on inverse problems in computational imaging.



**James Gregson** received the B.Eng. and M.A.Sc. degrees in mechanical engineering with a specialization in compressible multiphase flows. He is currently pursuing the Ph.D. degree with the University of British Columbia with a research on imaging inverse problems and their applications to fluid capture. He has worked in areas ranging from geometry processing to computational displays.

**Felix Heide** received the B.Sc. and M.Sc. degrees from the University of Siegen. He is currently pursuing the Ph.D. degree with the University of British Columbia. His research interests are centered on computational photography, optimization, and displays.

**Wolfgang Heidrich** received the Ph.D. degree in computer science from the University of Erlangen, in 1999. He is currently the Director of the Visual Computing Center with the King Abdullah University of Science and Technology. He is also affiliated with the University of British Columbia, where he held the Dolby Research Chair until 2013. He was a Research Associate with the Computer Graphics Group, Max-Planck-Institute for Computer Science, Saarbrucken, Germany, before joining UBC in 2000. In particular, he has worked on computational photography and displays, high dynamic range imaging and display, image-based modeling, measuring, rendering, geometry acquisition, GPU-based rendering, and global illumination. His work on high dynamic range displays served as the basis for the technology behind Brightside Technologies, which was acquired by Dolby in 2007. He has written well over 150 refereed publications on these subjects and has served on numerous program committees. His research interests lie at the intersection of computer graphics, computer vision, imaging, and optics. He was a recipient of the 2014 Humboldt Research Award.