Blind Deconvolution Using a Normalized Sparsity Measure

Dilip Krishnan
Courant Institute
New York University
dilip@cs.nyu.edu

Terence Tay
Chatham Digital
ttay@chathamdigital.com

Rob Fergus
Courant Institute
New York University
fergus@cs.nyu.edu

Abstract

Blind image deconvolution is an ill-posed problem that requires regularization to solve. However, many common forms of image prior used in this setting have a major drawback in that the minimum of the resulting cost function does not *correspond to the true sharp solution. Accordingly, a range of workaround methods are needed to yield good results (e.g. Bayesian methods, adaptive cost functions, alpha-matte extraction and edge localization). In this paper we introduce a new type of image regularization which gives lowest cost for the true sharp image. This allows a very simple cost formulation to be used for the blind deconvolution model, obviating the need for additional methods. Due to its simplicity the algorithm is fast and very robust. We demonstrate our method on real images with both spatially invariant and spatially varying blur.*

1. Introduction

Low-level vision tasks, such as denoising, deblurring, in-painting and super-resolution involve the recovery of a sharp distortion-free image from a corrupted and degraded input. As these problems are ill-posed, most approaches introduce an image prior that favors natural images over unnatural (i.e. noisy or blurry) ones. By regularizing the problem in this fashion, a high quality result can be recovered.

A wide range of parametric image priors have been proposed. The simplest are Gaussian smoothness penalties on the output of local derivative operators. But since images are highly non-Gaussian [5], many approaches [6, 13, 14] instead use ℓ_p-norms on the gradients, with $0.7 \leq p \leq 1$, reflecting the statistics of natural images. More sophisticated approaches try and learn the filters and/or energy functions. Zhu and Mumford [26] learn arbitrary energy functions for a set of oriented derivative filters via Gibbs sampling in a maximum likelihood approach. Roth and Black [21] introduce the Fields of Experts model that employs student-T potential functions and learn the filters using contrastive divergence. Weiss and Freeman [23] propose a simpler learning scheme for the Fields of Experts model that allows the efficient training of large filters. Raj and Zabih [20] propose a discrete Markov random field based smoothness prior that can efficiently be minimized using graph cuts.

While these models have been successfully used in tasks such as denoising and optical flow estimation, their application to blurry images is problematic. As noted by Levin *et al.* [15] and Fergus *et al.* [4], somewhat counter-intuitively, the above image models all favor blurry images to sharp images. In other words, blurry images have lower cost (are more probable) than sharp images. This is a direct result of the learned/chosen potential functions decreasing toward zero: since blur attenuates high frequencies, the response of any derivative-type filter will also be reduced and consequently will have a lower cost under the model. This phenomenon is illustrated in Fig. 1, where we show the cost (negative log probability) of a sharp image blurred by different amounts. As a consequence, for blind deconvolution, which is highly ill-posed and hence reliant on the image prior, standard maximum a-posterori (MAP) based approaches do not work. Correspondingly, a number of more complex methods have been proposed. These include: marginalization over all possible images [4, 15, 17]; dynamic adaptation of the cost function [22]; alpha-matt extraction [10]; re-weighting of the image edges [3]; determination of the edge locations using shock filtering [18].

In this paper we introduce a novel type of image regularization that favors sharp images over blurry and show how this prior can be used in a framework for blind deconvolution. Compared to other methods, our approach is very simple – it requires none of the complexities needed by other methods to overcome shortcomings of existing priors in an MAP setting. The resulting scheme is also quick since it can take advantage of existing fast ℓ_1 methods to estimate the kernel and sharp image.

2. Motivation

The regularization function we propose is the ratio of the ℓ_1 norm to the ℓ_2 norm on the high frequencies of an image. ℓ_1/ℓ_2 is an unusual function and its relevance to blind image deconvolution is not immediately clear. We first motivate its use in this setting, before explaining our method.

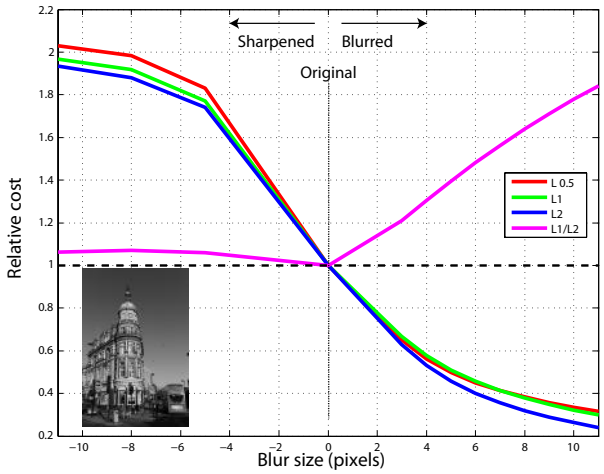

Figure 1. A comparison of our novel image regularizer to existing approaches. For a typical real world scene I (inset), we add Gaussian blur b ranging from 0 to 11 pixels in size and measure the cost: $\|(\nabla_x(I \otimes b))\|_\alpha + \|(\nabla_y(I \otimes b))\|_\alpha$ where ∇_x, ∇_y are the x and y derivatives respectively. Existing image regularizers use $\alpha = 0.5$ (red), 1 (green), 2 (blue). Our image regularizer $\frac{\|[\nabla_x(I \otimes b), \nabla_y(I \otimes b)]\|_1}{\|[\nabla_x(I \otimes b), \nabla_y(I \otimes b)]\|_2}$ is shown in magenta. The y-axis shows cost relative to that of the sharp image. A negative blur size corresponds to an unsharp mask filter. Note that the existing priors incorrectly have a *lower* cost for blurry images than sharp ones. By contrast, our regularizer correctly gives lowest cost to the original image.

First consider the ℓ_1 norm. The ℓ_1 norm is widely used to impose signal sparsity, but it is scale variant so the norm can be minimized by simply reducing the signal. In an image setting, the ℓ_1 norm is typically used to penalize the high frequency bands. As image noise presents itself in these bands, boosting their ℓ_1 norm, minimizing the norm is a way of denoising the image. However, in the case of image blur, the opposite situation holds since blur attenuates the high frequency bands so reducing their ℓ_1 norm. Consequently, in a blind deconvolution setting where the kernel is only loosely constrained, minimizing the ℓ_1 norm on the high frequencies of the image will result in a blurry image (and a delta function kernel). This behavior, studied in [15], is illustrated in Fig. 1.

The simplest interpretation of the ℓ_1/ℓ_2 function is that it is a normalized version of ℓ_1 , making it scale invariant. If applied to the high frequency bands of an image, it is equivalent to the ℓ_1 norm of the edges rescaled by their total energy. Although blur decreases both the ℓ_1 and ℓ_2 norms, crucially the latter is reduced more, thus the *ratio* of the two will be increased by blur (see the magenta curve in Fig. 1).

To understand why this is so, consider the visualization of the ℓ_1/ℓ_2 function for a two dimensional signal in Fig. 2. The minima lie along the axes with the cost increasing smoothly in between. The high frequency bands of natural scenes are typically sparse in that the magnitudes

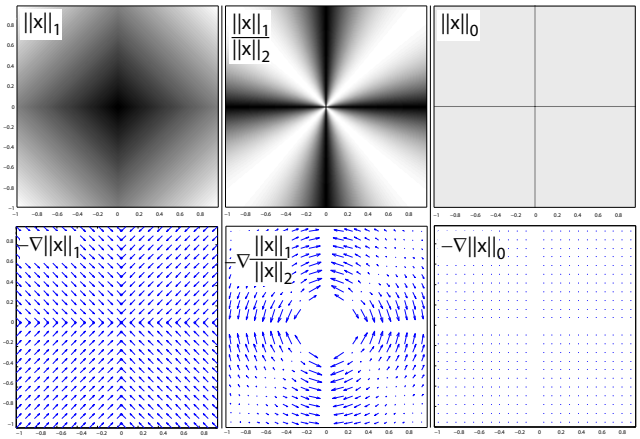

Figure 2. A visualization of ℓ_1 , ℓ_1/ℓ_2 and ℓ_0 functions (top row) and their negative gradient fields (bottom row) for a two dimensional vector (dark corresponds to a lower value). The l_1 function is smallest at the origin and the gradient field uniformly points towards the origin. The ℓ_1/ℓ_2 function has minima along the axes, with smoothly increasing cost in the diagonal directions. Its gradient field is purely radial: starting at arbitrary location the gradient leads to the nearest axis, preserving distance from the origin. Note that the minima structure of ℓ_1/ℓ_2 is very similar that ℓ_0 . However, the ℓ_0 norm is difficult to use, having zero gradient everywhere, except near the axes where it is infinite.

are mostly either zero or very small, but occasionally large. If these bands are represented as a single high dimensional vector, it would be close to the axes in many dimensions and have a low ℓ_1/ℓ_2 value. Blur smears out the large magnitude elements and reduces the number of zero elements in the vector, so rotating it diagonally away from the axes, increasing the ℓ_1/ℓ_2 value.

Given that the ℓ_1/ℓ_2 function behaves correctly for blur, it is natural to wonder if added noise or sharpening operations might result in a lower cost than the true image. Noise added to the signal increases the ℓ_1/ℓ_2 value, as do sharpening operations (see left side of Fig. 1).

Most of the energy in images is contained in the low and mid frequency bands, which are barely affected by blur. As a consequence, the ℓ_1/ℓ_2 function will not be changed significantly if measuring the entire image (i.e. all frequency bands). Instead, the ℓ_1/ℓ_2 function must be applied to just the high frequency part of the image if it is to discriminate between sharp and blurry images.

The ℓ_1/ℓ_2 function is one of a number of sparsity measures in the literature [9] but is relatively rare, being previously used for matrix factorization [8, 19]. In [9], different sparsity measures are compared using 6 heuristic criteria and the ℓ_1/ℓ_2 function satisfies all them.

Sparsity has a natural interpretation using an ℓ_0 measure. However, ℓ_0 is difficult to optimize because of the lack of gradient information everywhere. It is therefore convenient to use a convex measure such as ℓ_1 instead [2]. But, as illustrated in Fig. 2, ℓ_1 has a very different shape to ℓ_0 and it

is also not scale invariant. The ℓ_1/ℓ_2 function, on the other hand, is scale invariant and has minima along the axes, just as ℓ_0 does. It also has the advantage of gradient information which can be exploited to give a tractable optimization algorithm.

The ℓ_1/ℓ_2 function does have several drawbacks. First, it is non-convex, unlike ℓ_1 , thus there are multiple local-minima. Second, it is hard to optimize directly but we introduce approximate methods that can overcome this. Finally, it cannot be expressed as a probabilistic prior as $\int exp(-\|x\|_1/\|x\|_2)dx = \infty$, due to the scale invariance. This is in direct contrast to ℓ_p norms ($0.7 \leq p \leq 1$) which correspond to probabilistic models of image gradients, having a (hyper)-Laplacian form. However, since we intend to use a non-probabilistic framework, it does not matter that the energy surface is not normalized.

3. Approach

We assume the formation model of a sharp image u blurred by a matrix K along with the addition of Gaussian i.i.d noise N:

$$g = Ku + N \tag{1}$$

We observe the resulting blurry image g and our goal is to recover the unknown sharp image u and the blurring matrix K. Algorithm 1 outlines our approach.

Algorithm 1 : Overall Algorithm

Require: Observed blurry image g, Maximum kernel size h.
Apply derivative filters to g, creating a high-freq. image y.
1. Blind estimation of blur matrix K (Section 3.1) from y.
Loop over coarse-to-fine levels:
Alternate:
- Update sharp high-frequency image x (Section 3.1.1) using l_1/l_2 regularization.
- Update blurring matrix K (Section 3.1.2).
Interpolate solution to finer level as initialization.
2.Image recovery using non-blind algorithm of [12] (Section 3.2).
- Deblur g using K to give sharp image u.
return Sharp image u.

In Section 3.1, we first consider the case when the blur is spatially constant. In this case, the matrix K reduces to a 2-dimensional convolution operation with a kernel k. We then show how our algorithm can be extended to the case of pure in-plane rotation in Section 3.4. Finally, we consider the case of general 3-D rotations of the camera in Section 3.5. The overall algorithm is implemented in a multiscale framework, described in Section 3.1.3.

3.1. Blind Kernel Estimation

Our kernel estimation is performed on the high frequencies of the image. Given the blurry and noisy input g, we use discrete filters $\nabla_x = [1, -1]$ and $\nabla_y = [1, -1]^T$ to generate a high-frequency version $y = [\nabla_x g, \nabla_y g]$[1]. The cost function for spatially invariant blurring is:

$$\min_{x,k} \lambda\|x \otimes k - y\|_2^2 + \frac{\|x\|_1}{\|x\|_2} + \psi\|k\|_1 \tag{2}$$

subject to the contraints that $k \geq 0, \sum_i k_i = 1$. Here x is the unknown sharp image in the high-frequency space, k is the unknown blurring kernel (k_i are individual elements) and $\otimes$ is the 2D convolution operator.

Eqn. 2 consists of 3 terms. The first term is the likelihood that takes into account the formation model Eqn. 1. The second term is the new l_1/l_2 regularizer on x which encourages scale-invariant sparsity in the reconstruction. To reduce noise in the kernel, we add ℓ_1 regularization on k. The constraints on k (sum-to-1 and non-negativity) follow from the physical principles of blur formation. The scalar weights λ and ψ control the relative strength of the kernel and image regularization terms.

Eqn. 2 is highly nonconvex. The standard approach to optimizing such a problem is to start with an initialization on x and k, and then alternate between x and k updates [4]. To make consistent progress along each of the unknowns and avoid local minima as far as possible, only a few iterations are performed in each update.

3.1.1 x Update

The x sub-problem is given by:

$$\min_{x} \lambda\|x \otimes k - y\|_2^2 + \frac{\|x\|_1}{\|x\|_2} \tag{3}$$

This sub-problem is non-convex due to the presence of the new regularization term $\frac{\|x\|_1}{\|x\|_2}$. However, if one fixes the denominator of the regularizer from the previous iterate, the problem then becomes a convex l_1-regularized problem. Fast algorithms to solve l_1-regularized problems are well-known in the compressed sensing literature [1, 25]. One such algorithm is the iterative shrinkage-thresholding algorithm (ISTA) [1]. ISTA, detailed in Algorithm 2, is a fast method to solve general linear inverse problems of the form:

$$\min_{x} \lambda\|Kx - y\|_2^2 + \|x\|_1 \tag{4}$$

In our application K is the blurring matrix.

The operator S in Algorithm 2 is the soft shrinkage operation on a vector. It shrinks each component of the input vector towards zero:

$$S_\alpha(x)_i = \max(|x_i| - \alpha, 0)\text{sign}(x_i) \tag{5}$$

ISTA is very simple and fast, involving only multiplications of the matrix K with vector x, followed by the component-wise shrinkage operation.

We use the ISTA step as the inner iteration in our x-update algorithm. The outer loop then simply re-estimates

[1] y is a concatenation of the two gradient images $\nabla_x g$ and $\nabla_y g$.

Algorithm 2 : Iterative Shrinkage-Thresholding Algorithm (ISTA)

Require: Operator K, regularization parameter λ
Require: Initial iterate x^0, observed image y
Require: Threshold t, maximum iterations N
1: **for** $j = 0$ to $N - 1$ **do**
2: $\quad v = y - tK^T(Kx^j - y)$
3: $\quad x^{j+1} = S_{t\lambda}(v)$
4: **end for**
5: **return** Output image x^N

the weighting on the likelihood term in Eqn. 3 by updating the denominator $\|x\|_2$. The overall x-update algorithm is as follows:

Algorithm 3 : x Update

Require: Blur kernel k from previous k update
Require: Image x^0 from previous x update
Require: Regularization parameter $\lambda = 20$
Require: Maximum outer iterations $M = 2$, inner its. $N = 2$
Require: ISTA threshold $t = 0.001$
1: **for** $j = 0$ to $M - 1$ **do**
2: $\quad \lambda' = \lambda \|x^j\|_2$
3: $\quad x^{j+1} = \text{ISTA}(k, \lambda', x^j, t, N)$
4: **end for**
5: **return** Updated image x^M.

Despite the non-convexity of the problem, in practice this inner-outer iteration is effective in reducing the cost function in Eqn. 3. After an x-update step, we update the kernel estimate k.

3.1.2 k Update

The kernel update sub-problem is given by:

$$\min_k \lambda \|x \otimes k - y\|_2^2 + \psi \|k\|_1 \tag{6}$$

subject to the constraints $k \geq 0, \sum_i k_i = 1$. We use unconstrained iterative re-weighted least squares (IRLS) [13] followed by a projection of the resulting k onto the contraints (setting negative elements to 0, and renormalizing). During the iterations we run IRLS for just 1 iteration, with the weights being computed from the kernel of the previous k update. We solve the inner IRLS system to a low level of accuracy, using a few (less than 5) conjugate gradient (CG) iterations.

An important practical point is that after recovering the kernel at the finest level, we threshold small elements of the kernel to zero, thereby increasing robustness to noise. This is similar to other blind deconvolution methods [4, 24].

3.1.3 Multiscale Implementation

For large kernels, an excessive number of x and k updates may be required to converge to a reasonable solution. To mitigate this problem, we perform multiscale estimation of the kernel using a coarse-to-fine pyramid of image resolutions, in a similar manner as in [4]. We use levels with a size ratio of $\sqrt{2}$ between them (in each dimension). The number of levels is determined by the size of the kernel K such that the kernel size at the coarsest level is 3×3. We downsample the input blurry image and then take discrete gradients to form the input y each level.

At each scale level we perform 200 alternating updates of x and k. Once a kernel estimate k and sharp gradient image x are computed, they are upsampled to act as the initialization of the kernel and sharp image at the next finer level. All of the resizing operations are done using bilinear interpolation.

3.2. Image Recovery

Once the kernel k for the finest level has been estimated, we can use a variety of non-blind deconvolution methods to recover the full-spectrum sharp image u from g. The simplest is Richardson-Lucy (RL). The disadvantage of RL is that this method is sensitive to a wrong kernel estimate, which results in ringing artifacts in u. Therefore, we choose to use the non-blind deconvolution method from [12], since it is fast and robust to small kernel errors. This algorithm uses a continuation method to solve the following cost function:

$$\min_u \lambda \|u \otimes k - g\|_2^2 + \|\nabla_x g\|_\alpha + \|\nabla_y g\|_\alpha \tag{7}$$

where ∇_x and ∇_y are the same derivative filters used in Section 3.1. We use $\lambda = 3000, \alpha = 0.8$ for all results.

We did not use the ℓ_1/ℓ_2 regularization term for the image recovery. This is because the non-blind deconvolution problem is much less ill-posed than blind deconvolution; and l_p-type regularizers such as those used in [12] work well and are fast.

3.3. Speed and Robustness

Our cost function Eqn. 2 is of a simple form. The x and k update steps involve a few matrix-vector (or convolution) operations. As a result, our algorithm is quite fast as compared to existing algorithms such as [4]. For a 255×255 pixel image, and when estimating a 35×35 size kernel, our algorithm takes 3 minutes, compared to 6 minutes for the method of [4].[2] Our method is amenable to speedups such as GPU acceleration and the use of the Intel IPP libraries.

In our experiments, we find the algorithm to be robust to the choice of parameters and use the same settings for all results reported in this paper. The ψ parameter depends on the user-specified kernel size h, according to the formula: $\psi = \frac{3}{13}h$. This robustness is a major advantage of our algorithm over existing schemes that require parameter adjustment for different input images. Empirically, we also find

[2] Using a single-threaded Matlab on a 2.66Ghz CPU.

that in all our test cases, the function value in Eqn. 2 reduces monotonically as the iterations proceed.

3.4. Extension to In-Plane Rotation

We extend the cost function of Eqn. 2 to the case where the blurring process arises purely from rotation of the camera around the Z-axis (axis perpendicular to the sensor plane), thus is no longer spatially constant. Similar to the non-uniform blurring model developed in [24], we assume that the blurred image arises form a linear combination of discretely sampled rotations of the sharp image:

$$g = \sum_{\theta \in \omega} k_\theta R_\theta(u) + N \tag{8}$$

where ω is a discrete set of angles, and the operator $R_\theta(u)$ rotates the image u by angle θ. For this formation model our blur kernel is the vector k with entries k_θ. The minimization problem is then a modified version of Eqn. 2, given by:

$$\min_{x,k} \lambda \| \sum_{\theta \in \omega} k_\theta R_\theta(x) - y \|_2^2 + \frac{\|x\|_1}{\|x\|_2} + \psi \|k\|_1 \tag{9}$$

subject to the constraints that $k \geq 0$ and $\sum_{\theta \in \omega} k_\theta = 1$. We can now adopt exactly the same approach as for the spatially invariant case, except replacing 2-dimensional convolutional operations with sums of rotations.

3.5. Extension to 3-D Rotations

By extending the set of rotations ω to the X and Y axes, our model can be generalized to the full non-uniform blur model. It has been demonstrated in [24] that rotations of the camera about X and Y cause more significant blur than translations. Accordingly, we adopt the model proposed in [24] that samples discretely from all 3 planes of rotation. The final formation model and cost function formulation are analogous to Eqn. 8 and Eqn. 9:

$$\min_{x,k} \lambda \| \sum_{\theta_{xyz} \in \omega} k_\theta R_{\theta_{xyz}}(x) - y \|_2^2 + \frac{\|x\|_1}{\|x\|_2} + \psi \|k\|_1 \tag{10}$$

where θ_{xyz} refers to a particular combination of rotations around all 3 axes. ω is a set of discrete angles of rotation about each axis. For spatially varying kernels arising from camera rotations, the non-blind deconvolution algorithm must be modified to take into account the different formation model. We extend the non-blind deconvolution algorithm of [12] accordingly.

Gupta *et al.* [6] present a blur formation model based on in-plane rotations combined with translations. They also use a set of discretized basis functions. By replacing the basis functions $R_{\theta_{xyz}}$ in Eqn. 10 with their basis functions, we can support their model with our new regularizer. Finally, our regularization scheme fits into the efficient spatially variant model of [7].

4. Experiments

In this section, we present results of our algorithm and compare it to the algorithms of [4],[22] and [24]. We consider spatially invariant, pure in-plane rotation and 3-D rotational blurring models. We show results on both synthetic and real-world examples. The images are best viewed on screen.

4.1. Spatially Invariant Kernel

4.1.1 Synthetic Data

We first test the algorithm on the spatially invariant kernels from the dataset in Levin *et al.* [15]. This dataset consists of 4 images of size 255×255 and 8 different kernels ranging in size from 13×13 to 27×27 to give a total of 32 blurred images. The blurred data, ground truth data and ground truth kernels are provided. We compare our kernel estimation results with the blind deconvolution algorithms of Fergus *et al.* [4] and Shan *et al.* [22]. For kernels estimated by the 3 different methods, we perform non-blind deconvolution using the algorithm of [12] with the same parameter settings. The error metric used is the same as [15][3].

In Fig. 3, we plot the cumulative histograms of the error ratios for the 3 algorithms. The performance of our algorithm is similar to that of Fergus *et al.* [4]. Both these algorithms significantly out perform that of Shan *et al.* [22]. Additionally, there are a few examples in the dataset (see [16] for details) when the algorithm of [4] fails dramatically, whereas our algorithm is still able to recover a reasonable kernel. In addition, our algorithm is considerably simpler and faster than that of [4].

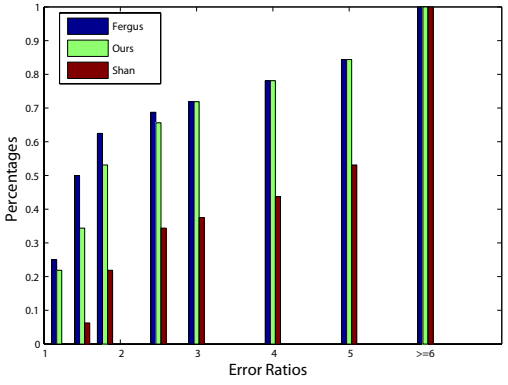

Figure 3. Cumulative histograms of the error ratios across the dataset of Levin *et al.* [15]. See text for details.

Fig. 4 shows results with a kernel size 27×27. In Fig. 5, we show a failure case for both [4] and [22]. By constrast,

[3]The measure is the *ratio* of SSD (sum of squared differences) error between: (i) the deconvolved output and the known ground truth image and (ii) deconvolved output using the ground truth kernel and the ground truth image.

Figure 4. Recovery of a 27×27 kernel. Top-left: original; top-right: blurred; middle-left: deblurred with ground truth kernel; middle-right: deblurred with our estimated kernel; bottom-left: deblurred with kernel of [4]; bottom-right: deblurred with kernel of [22]. Corresponding kernels are shown as insets at the bottom left of each image.

our algorithm is able to recover a reasonable kernel. This illustrates the robustness provided by our new l_1/l_2 regularization function.

4.1.2 Real Data

Next we compare the performance of our algorithm on some real-world examples presented in different blind deconvolution papers. As before, we use the different blind algorithms to estimate the kernel and then use the same non-blind algorithm (that of [12]) with identical parameter settings to perform the deconvolution. Fig. 6 shows the kernel recovery and reconstruction for an image provided in the online code of Fergus *et al.* [4], along with their result. We also compare with the result by running code provided to us by Cho *et al.* [3]. The result of [3] shows artefacts, for example, in the cheeks of the statue. Fig. 7 compares kernel recovery and reconstruction of another image from [4]. Our results were obtained using the same parameter settings as

Figure 5. Recovery of a 27×27 kernel. Top-left: original; top-right: blurred; middle-left: deblurred with ground truth kernel; middle-right: deblurred with our estimated kernel; bottom-left: deblurred with kernel of [4]; bottom-right: deblurred with kernel of [22].

for the synthetic data.

4.2. In-Plane Rotation

We now take a synthetic example presented in [24] of pure in-plane rotation of the camera. We use the model of Eqn. 8 to perform the kernel estimation. In this case, we use only a single scale of processing (the finest scale) to speed up the computation. The other settings such as the number of iterations, and regularization parameter λ remain unchanged from the spatially invariant setting. We deblur the image (with 2% noise added) with both our algorithm and that of [24]. The results are shown in Fig. 8. The method of [24] took over 3 hours for the entire processing, whereas our method takes 10 minutes on the same CPU.

4.3. 3-D Rotation

In Fig. 10, we compare our kernel estimation on a real world example given in [24]. The ground truth kernel is unknown in this case. We use the model of Eqn. 10 with the same discretized set of 3D rotation angles as used in the

Figure 6. Recovery of a real-world kernel. Top-left: Input blurry image; top-right: deblurred with kernel of [4]; bottom-left: deblurred with our estimated kernel; bottom-right: deblurred with algorithm of [3]. Recovered kernels (of size 25×25) are shown as insets.

online code of [24]. Our results are very similar to those of [24]. In Fig. 9, we also use the same set of rotation angles to deblur a cropped section of an image from [11].

5. Discussion

Our approach to blind deconvolution is motivated by a fundamental re-analysis of the interaction between image regularizers and the effects of blur on the high frequencies in an image. The crucial component of our algorithm is the introduction of a novel scale-invariant regularizer that compensates for the attenuation of high frequencies and therefore greatly stabilizes the kernel estimation process. However, since this regularizer is non-convex, we introduce a minimization scheme that amounts to solving a series of l_1 problems with different regularization parameters. The resulting algorithm is applicable to different models of blur formation and is fast and robust to the choice of parameters. Matlab code for our algorithm, images and supplementary material are available at `www.cs.nyu.edu/~dilip/wordpress/?page_id=159`.

Figure 7. Recovery of a real-world kernel. Top-left: Input blurry image; top-right: deblurred with kernel of [4]; bottom-left: deblurred with our estimated kernel; bottom-right: deblurred with algorithm of [3]. Recovered kernels (of size 25×25) are shown as insets.

Figure 8. In-plane rotational deblurring. Top-left: input sharp image; top-right: blurry image with pure in-plane rotation; bottom-left: deblurred with code of [24]; bottom-right: our deblurred result.

Acknowledgements

Dilip Krishnan is supported by a Microsoft Graduate Fellowship. Rob Fergus thanks the DARPA Deep Learning and ONR Information Integration programs for support. The authors would also like to thank Fredo Durand, Bill Freeman and Anat Levin for helpful discussions.

References

[1] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. In *SIAM J. on Img. Sciences*, pages 183–202, 2009.

Figure 10. 3D rotational deblurring. Left: Input blurry image; middle: deblurred with algorithm of [24]; right: deblurred with our algorithm.

Figure 9. 3D rotational deblurring. Left: Input blurry image (cropped version of image from [11]); right: deblurred with our algorithm.

[2] E. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inform. Theory*, 52:489–509, 2004.

[3] S. Cho and S. Lee. Fast motion deblurring. *SIGGRAPH ASIA 2009*, 28(5), 2009.

[4] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. Freeman. Removing camera shake from a single photograph. *SIGGRAPH*, 25:787–794, 2006.

[5] D. Field. What is the goal of sensory coding? *Neural Computation*, 6:559–601, 1994.

[6] A. Gupta, N. Joshi, L. Zitnick, M. Cohen, and B. Curless. Single image deblurring using motion density functions. In *ECCV '10*, 2010.

[7] M. Hirsch, S. Sra, B. Scholkopf, and S. Harmeling. Efficient filter flow for space-variant multiframe blind deconvolution. In *CVPR*, pages 607–614, 2010.

[8] P. Hoyer. Non-negative matrix factorization with sparseness constraints. *JMLR*, 5:1457–1469, 2004.

[9] N. Hurley and S. Rickard. Comparing measures of sparsity. *IEEE Trans. Inf. Theory*, pages 4723–4741, 2009.

[10] J. Jia. Single image motion deblurring using transparency. In *CVPR*, 2007.

[11] N. Joshi, S. B. Kang, C. L. Zitnick, and R. Szeliski. Image deblurring using inertial measurement sensors. *SIGGRAPH*, 2010.

[12] D. Krishnan and R. Fergus. Fast image deconvolution using hyper-laplacian priors. In *NIPS*, 2009.

[13] A. Levin, R. Fergus, F. Durand, and W. Freeman. Image and depth from a conventional camera with a coded aperture. *SIGGRAPH*, 26(3):70, 2007.

[14] A. Levin and Y. Weiss. User assisted separation of reflections from a single image using a sparsity prior. *PAMI*, 29(9):1647–1654, Sept 2007.

[15] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Understanding and evaluating blind deconvolution algorithms. In *CVPR*, 2009.

[16] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Understanding and evaluating blind deconvolution algorithms. In *Extended Technical Report*, 2009.

[17] J. Miskin and D. J. C. MacKay. Ensemble Learning for Blind Image Separation and Deconvolution. In M. Girolani, editor, *Adv. in Independent Component Analysis*. Springer-Verlag, 2000.

[18] J. Money and S. Kang. Total variation minimizing blind deconvolution with shock filter reference. In *Image and Vision Computing*, number 2, pages 302–314, 2008.

[19] M. Mørup, K. H. Madsen, and L. K. Hansen. Approximate l_0 constrained non-negative matrix and tensor factorization. In *ISCAS 2008 special session on Non-negative Matrix and Tensor Factorization and Related Problems*, 2008.

[20] A. Raj and R. Zabih. A graph-cut problem for general deconvolution problems. In *CVPR*, 2005.

[21] S. Roth and M. J. Black. Fields of Experts: A Framework for Learning Image Priors. In *CVPR*, volume 2, pages 860–867, 2005.

[22] Q. Shan, J. Jia, and A. Agarwala. High quality motion deblurring from a single image. *SIGGRAPH*, 27, 2008.

[23] Y. Weiss and W. T. Freeman. What makes a good model of natural images? In *CVPR*, 2007.

[24] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce. Non-uniform deblurring for shaken images. In *CVPR*, 2010.

[25] W. Yin, S. Osher, D. Goldfarb, and J. Darbon. Bregman iterative algorithms for l1-minimization with applications to compressed sensing. *SIAM J. on Img. Sciences*, 1(1):143–168, 2008.

[26] S. C. Zhu and D. Mumford. Prior learning and Gibbs reaction-diffusion. *IEEE Trans. PAMI*, 19(11):1236–1250, 1997.