Efficient, Blind, Spatially-Variant Deblurring for Shaken Images

Oliver Whyte, Josef Sivic, Andrew Zisserman, Jean Ponce

1 Efficient, Blind, Spatially-Variant Deblurring for Shaken Images

Oliver Whyte, Josef Sivic, Andrew Zisserman and Jean Ponce

Abstract

In this chapter we discuss modeling and removing spatially-variant blur from photographs. We describe a compact global parameterization of camera shake blur, based on the 3D rotation of the camera during the exposure. Our model uses three-parameter homographies to connect camera motion to image motion and, by assigning weights to a set of these homographies, can be seen as a generalization of the standard, spatially-invariant convolutional model of image blur. As such we show how existing algorithms, designed for spatially-invariant deblurring, can be "upgraded" in a straightforward manner to handle spatially-variant blur instead. We demonstrate this with algorithms working on real images, showing results for blind estimation of blur parameters from single images, followed by non-blind image restoration using these parameters. Finally, we introduce an efficient approximation to the global model, which significantly reduces the computational cost of modeling the spatially-variant blur. By approximating the blur as locally-uniform, we can take advantage of fast Fourier-domain convolution and deconvolution, reducing the time required for blind deblurring by an order of magnitude.

1.1 Introduction

Everybody is familiar with camera shake, since the resulting blur spoils many photos taken in low-light conditions. Camera shake blur is caused by motion of the camera during the exposure; while the shutter is open, the camera passes through a sequence of different poses, each of which gives a different view of the scene. The sensor accumulates all of these views, summing them up to form the recorded image, which is blurred as a result. We would like to be able to deblur such images to recover the underlying sharp image, which we would have captured if the camera had not moved.

In general, the problem of restoring an image after it has suffered some degradation can be broken down into three stages: first, a generative model is needed to relate the undegraded, "ideal" or latent image (that we would like to recover) to the observed image produced by the camera. Second, the parameters of this

model must be estimated, and finally the restored image can be reconstructed, given the model and the estimated parameters.

This chapter is principally concerned with the first of these stages: a geometrically motivated model of *spatially-variant* image blur due to camera shake, which we show can be (mostly) attributed to the rotation of the camera during exposure. We develop a global descriptor for the generative model parameters of this non-uniform blur, analogous to (but different from) a convolution kernel, and show that a more general class of blurs can be modelled than uniform.

Several authors have proposed models for spatially-variant blur, under different assumptions about the scene and the camera, e.g. simple scene models with unconstrained camera motion (Joshi, Kang, Zitnick & Szeliski 2010, Gupta, Joshi, Zitnick, Cohen & Curless 2010, Tai, Tan & Brown 2011), constrained camera motion (Sawchuk 1974, Klein & Drummond 2005, Shan, Xiong & Jia 2007, Tai, Kong, Lin & Shin 2010), or more complex scene models (Šorel & Flusser 2008, Xu & Jia 2012). On the other hand, much of the work on *algorithms* for deblurring camera shake assume that the blurred image is simply a 2D convolution of a sharp image with a *spatially-invariant* filter (Fergus, Singh, Hertzmann, Roweis & Freeman 2006, Shan, Jia & Agarwala 2008, Cho & Lee 2009), despite the fact that real camera shake does not, in general, cause spatially-invariant blur (Levin, Weiss, Durand & Freeman 2009), as shown in Figure 1.1. In this chapter we aim to bridge this gap, with a practical model for spatially-variant camera shake blur that can also leverage advances in convolution-based deblurring algorithms.

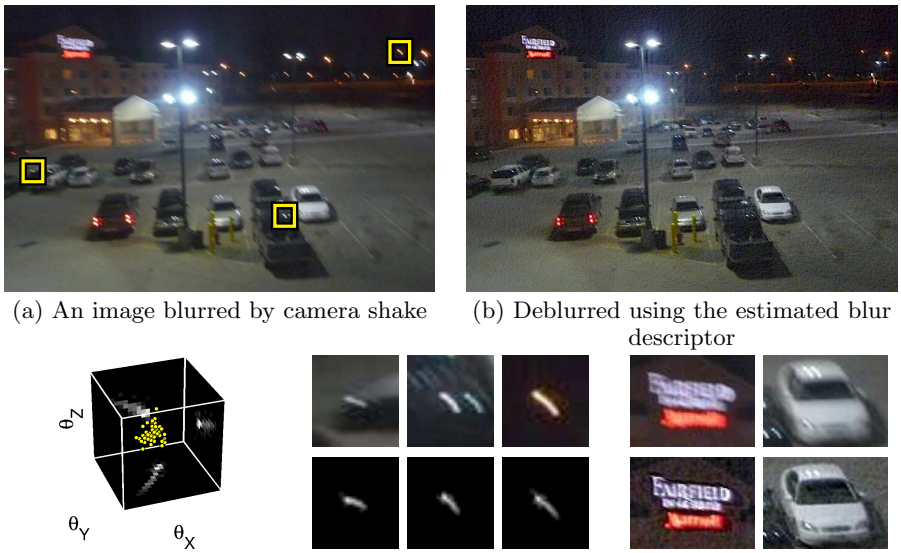
We begin in Section 1.2 by deriving our geometric model, before discussing how it can be implemented in practice in Section 1.3. In Section 1.4 we show how our model can replace convolution in existing deblurring algorithms, with minimal algorithmic changes. In Section 1.5 we describe an efficient approximation to the model, which significantly reduces its computational cost. In Section 1.6 we present results on real images, using our model to replace the uniform (convolution) blur model in an existing algorithm for camera shake removal. In Section 1.7 we cover some implementation considerations for our model.

Parts of this chapter are based on previously-published work (Whyte, Sivic, Zisserman & Ponce 2010, Whyte, Sivic & Zisserman 2011, Whyte, Sivic, Zisserman & Ponce 2012).

1.2 Modelling spatially-variant camera shake blur

A camera may move in several different ways, and it is not necessarily obvious which kinds of motion cause large changes in the view (and hence a large blur), and which cause relatively small changes. Furthermore, even if the camera's motion is fully known for a given photograph, a model is needed to translate this physical 3D motion into image-domain motion before we can begin to deblur the photograph.

We begin by considering the relative blurring effect of different camera mo-



(a) An image blurred by camera shake

(b) Deblurred using the estimated blur descriptor

(c) Estimated global blur descriptor

(d) Top: Details from blurred image. Bottom: corresponding local PSFs generated by (c)

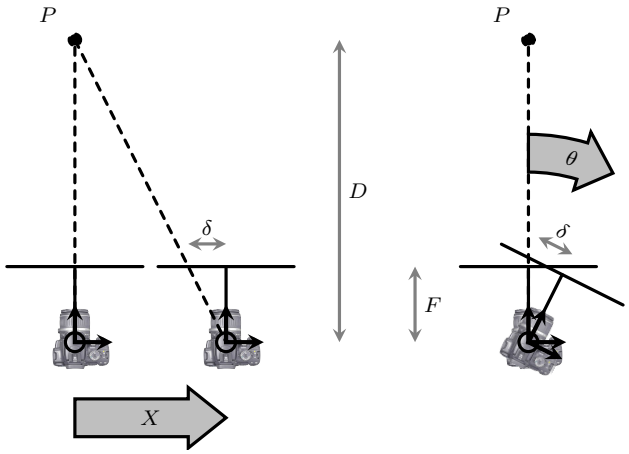
(e) Details from the blurred (top) and deblurred (bottom) images

Figure 1.1 Modelling non-uniform blur in a shaken image. The blurred image (a) clearly exhibits blur which is non-uniform, as highlighted at different locations in the image. Using the model proposed in this work, we can describe this blur using a single global descriptor (c), which in this case has been estimated from the blurred image itself, simply by modifying existing algorithms for blind deblurring (see Section 1.4 for details). Having estimated the blur, the standard Richardson-Lucy algorithm is used to estimate the sharp image. Close-ups of different parts of the image (d) show the variation in the shape of the blur, which can be accurately reproduced using our model, as shown by the local point spread functions generated from it. As can be seen in the deblurred image in (b) and the close-ups in (e), different parts of the image, blurred in different ways, can be deblurred to recover a sharp image. Reproduced from (Whyte et al. 2012) with permission, ©Springer 2012.

tions, and deriving a geometric model for camera shake. In Section 1.3 we develop this into a practical model for deblurring real images. In this work we limit our scope to photographs of static scenes, i.e. the blur is solely due to the motion of the camera.

1.2.1 Components of camera motion

The pose of a camera incorporates two components: *position* and *orientation*. Intuitively, the position tells us where the camera is, while the orientation tells us which way it is pointing, and both may vary while the camera's shutter is open. In this section, we discuss the contribution of each component to the image blur, and conclude that in most cases of camera shake, the changes in orientation



(a) Camera translation (b) Camera rotation

Figure 1.2 Blur due to translation or rotation of the camera. In this simplified example, we consider capturing a blurred image by either (a) translating the camera through a distance X parallel to the image plane, or (b) rotating the camera through an angle θ about its optical centre. We consider the scene point P at a distance D from the camera, whose image is blurred by δ pixels as a result of either of the two motions. In most cases, for a given blur size δ the rotation θ constitutes a significantly smaller motion of the photographer's hands than the translation X (see text for details). Reproduced from (Whyte et al. 2012) with permission, © Springer 2012.

(rotation) of the camera during exposure have a significantly larger blurring effect than the changes in position (translation).

Consider the simplified case shown in Figure 1.2 of a scene point P, at a distance D from the camera, being imaged at the centre of the camera's retina / sensor. During the exposure, the camera moves, and the image of the point is blurred through a distance δ pixels. In (a) the camera translates through a distance X parallel to the image plane, while in (b) the camera rotates through an angle θ about its optical centre. By simple trigonometry, we can see that in (a) the camera must translate by

$$X = \frac{\delta}{F} D, \tag{1.1}$$

where F is the camera's focal length, while in (b) the camera must rotate through an angle

$$\theta = \tan^{-1}\left(\frac{\delta}{F}\right). \tag{1.2}$$

If we make the common assumption that the camera's focal length F is approximately equal to the width of the sensor, say 1000 pixels, then to cause a blur of $\delta = 10$ pixels by translating the camera, we can see from Equation (1.1) that $X = \frac{1}{100} D$. Thus the required translation grows with the subject's distance

from the camera, and for a subject just 1 metre away, we must move the camera by $X = 1\,\text{cm}$ to cause the blur. When photographing a subject 30 metres away, such as a large landmark, we would have to move the camera by 30 cm!

To cause the same amount of blur by rotating the camera, on the other hand, we can see from Equation (1.2) that we would need to rotate the camera by $\theta = \tan^{-1}\left(\frac{1}{100}\right) \simeq 0.6°$, independent of the subject's distance from the camera. To put this in terms of the motion of the photographer's hands, then for example if the camera body is 10 cm wide, such a rotation could be caused by moving one hand just 1 mm forwards or backwards relative to the other. Provided the subject is more than 1 metre from the camera, this motion is at least an order of magnitude smaller than for a translation of the camera causing an equivalent amount of blur.

In reality, both the position and orientation of the camera vary simultaneously during the exposure. However, if the camera only undergoes small changes in position (translations), then following the discussion above, we can assert that the variations in the camera's orientation (rotations) are the only significant cause of blur.

From now on, we assume that the translational component of camera motion *does not cause any blur*. Furthermore, we assume that all rotations occur about the camera's optical centre. Note, however, that a camera rotation about a centre that is not the optical centre can be written as a rotation about the optical centre composed with a translation; these translations will generally be small if the centre of rotation is not far from the optical centre. As we shall see in the following section, this model of camera motion leads to spatially-variant blur.

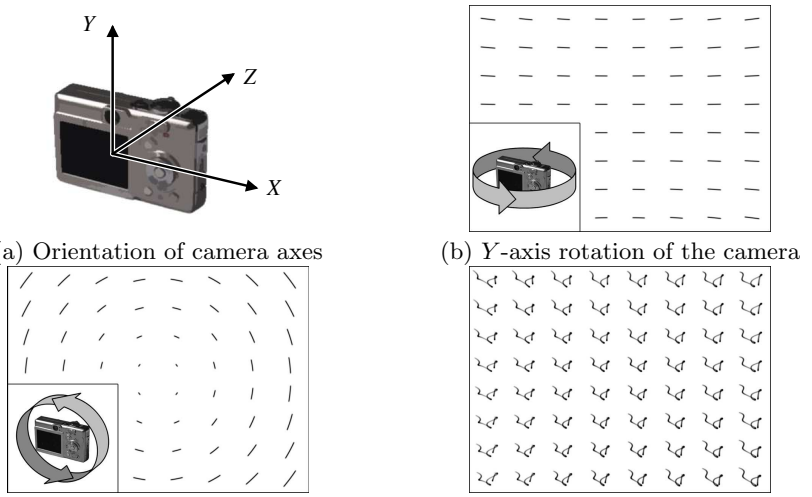
1.2.2 Motion blur and homographies

Under a pinhole camera model, and assuming that the scene being photographed is static, rotations of a camera about its optical centre induce projective transformations of the image being observed. In other words, the image observed at one camera orientation is related to the image at any other by a 2D projective transformation, or homography. For an uncalibrated camera, this is a general 8-parameter homography, but for a camera with known internal parameters, the homography $\mathbf{H}$ is specified by three parameters and is given by

$$\mathbf{H} = \mathbf{K}\mathbf{R}\mathbf{K}^{-1}, \tag{1.3}$$

where the 3×3 matrix $\mathbf{R}$ is a rotation matrix describing the orientation of the camera, and $\mathbf{K}$ is the camera's internal calibration matrix (Hartley & Zisserman 2004). In this work, we assume that the calibration matrix $\mathbf{K}$ is known (see Section 1.2.3).

The rotation matrix $\mathbf{R}$ has only three parameters. We adopt here the "angle-axis" parameterisation, in which a rotation is described by the angle θ moved about an axis $\mathbf{a}$ (a unit-norm 3-vector). This can be summarised in a single



(a) Orientation of camera axes (b) Y-axis rotation of the camera

(c) Z-axis rotation of the camera (d) Arbitrary sequence of rotations

Figure 1.3 Our coordinate frame with respect to initial camera orientation, and the paths followed by image points under different camera rotations. We define our coordinate frame (a) to have its origin at the camera's optical centre, with the X and Y axes aligned with those of the camera's sensor, and the Z axis parallel to the camera's optical axis. Under single-axis rotations of the camera, for example about its Y-axis (b), or its Z-axis (c), the paths traced by points in the image are visibly curved and non-uniform across the image. This non-uniformity remains true for general camera shakes (d), which do not follow such simple single-axis rotations, but rather take arbitrary paths through camera pose space. The focal length of the camera in this simulation is equal to the width of the image, the principal point is at the image's centre, and the pixels are assumed to be square. Reproduced from (Whyte et al. 2012) with permission, © Springer 2012.

3-vector $\boldsymbol{\theta} = \theta\mathbf{a} = (\theta_X, \theta_Y, \theta_Z)$. $\mathbf{R}$ is then given by the matrix exponential

$$\mathbf{R}_{\boldsymbol{\theta}} = e^{[\boldsymbol{\theta}]_\times}, \quad \text{where} \tag{1.4}$$

$$[\boldsymbol{\theta}]_\times = \begin{bmatrix} 0 & -\theta_Z & \theta_Y \\ \theta_Z & 0 & -\theta_X \\ -\theta_Y & \theta_X & 0 \end{bmatrix}. \tag{1.5}$$

We fix our 3D coordinate frame to have its origin at the camera's optical centre. The axes are aligned with the camera's initial orientation, such that the XY-plane is aligned with the camera sensor's coordinate frame and the Z-axis is parallel to the camera's optical axis, as shown in Figure 1.3 (a). In this configuration, θ_X describes the "pitch" of the camera, θ_Y the "yaw", and θ_Z the "roll", or in-plane rotation, of the camera.

Having defined the type of image transformations we expect to occur while the shutter is open, we can write out the image degradation model. Let T denote the exposure time of the photograph. While the shutter is open, the camera passes through a sequence of orientations $\boldsymbol{\theta}_t$, $t \in [0, T]$. As discussed above, at each

pose $\boldsymbol{\theta}_t$, the sensor is exposed to a projectively transformed version of the sharp image f, where the projective transformation $\mathbf{H}_t$ is given by Equations (1.3) to (1.5). The noiseless blurred image g^* is then modelled as the integral over the exposure time T of all the transformed versions of f:

$$g^*(\mathbf{x}) = \int_0^T f\big(\mathbf{H}_t\mathbf{x}\big)\,\mathrm{d}t, \tag{1.6}$$

where, with a slight abuse of notation, we use $g^*(\mathbf{x})$ to denote the value of g^* at the 2D image point represented by the homogeneous vector $\mathbf{x}$, and similarly for f.

Under this model, the apparent motion of scene points may vary significantly across the image. Figure 1.3 demonstrates this, showing the paths followed by points in an image under a Y-axis rotation, a Z-axis rotation, or an arbitrary sequence of rotations of the camera. Under the (in-plane) Z-axis rotation, the paths vary significantly across the image. Under the (out-of-plane) rotation about the Y-axis, the paths, while varying considerably less, are still non-uniform. It should be noted that the degree of non-uniformity of this out-of-plane motion is dependent on the focal length of the camera, decreasing as the focal length increases. However, it is typical for consumer cameras to have focal lengths of the same order as their sensor width, as is the case in Figure 1.3. In addition, it is common for camera shake to include an in-plane rotational motion. From this, it is clear that modelling camera shake as a convolution with a spatially-invariant kernel is insufficient to fully describe its effects (see also Figure 1.1).

In general, a blurred image has no temporal information associated with it, so it is convenient to replace the temporal integral in Equation (1.6) by a weighted integral over a set of camera orientations:

$$g^*(\mathbf{x}) = \int f\big(\mathbf{H}_{\boldsymbol{\theta}}\mathbf{x}\big)w(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta}, \tag{1.7}$$

where the weight function $w(\boldsymbol{\theta})$ encodes the camera's trajectory in a time-agnostic fashion. The weight will be zero everywhere except along the camera's trajectory, and the value of the function at a point $\boldsymbol{\theta}$ along the trajectory corresponds to the duration the camera spent at the orientation $\boldsymbol{\theta}$.

1.2.3 Camera calibration

In order to compute the homography in Equation (1.3) that is induced by a particular rotation of the camera, we need to know the camera's calibration matrix $\mathbf{K}$. For the results shown in this chapter, we assume that $\mathbf{K}$ takes the standard form

$$\mathbf{K} = \begin{bmatrix} F & 0 & x_0 \\ 0 & F & y_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{1.8}$$

This corresponds to a camera whose sensor has square pixels, and whose optical axis intersects the sensor at (x_0, y_0), referred to as the principal point. We recover the focal length of the camera from the image's EXIF tags, and assume that the principal point is at the centre of the image (which is typically sufficient for modelling homographies caused by camera rotations (Szeliski 2004)).

The radial distortion present in many digital cameras can represent a significant deviation from the pinhole camera model. Rather than modelling the distortion explicitly, we pre-process images with the commercially available PTLens software[1] to remove it. A second distortion present in many digital images comes from a non-linear intensity mapping applied by the camera storing the image, sometimes referred to as "gamma correction". In this work, we either use raw camera output images which do not have this non-linearity, or preprocess the blurred images with the inverse mapping.

1.2.4 Uniform blur as a special case

One consequence of our model for camera shake is that it includes spatially-invariant blur as a special case, and thus gives the conditions under which such a blur model is applicable. Given the definitions of $\mathbf{R}$, $\mathbf{K}$ and $\mathbf{H}$ in Equations (1.3), (1.4) and (1.8), and assuming $\theta_Z = 0$, it can be shown (Whyte 2012) that for small θ_X, θ_Y, as $F \to \infty$,

$$\mathbf{H}_{\boldsymbol{\theta}} \to \begin{bmatrix} 1 & 0 & F\theta_Y \\ 0 & 1 & -F\theta_X \\ 0 & 0 & 1 \end{bmatrix}, \tag{1.9}$$

which simply amounts to a translation of the image. Thus we can see that if the focal length of the camera is large (e.g. if the camera is zoomed-in) and there is no in-plane rotation, the blur which results from camera motion will be approximately spatially-invariant.

1.3 The computational model

Real cameras are equipped with a discrete set of pixels, and output a discrete set of samples of the degraded image, denoted by the vector $\mathbf{g} \in \mathbb{R}_+^N$, where $N = H \times W$ pixels for an image with H rows and W columns. We consider the sharp image also to be discrete: $\mathbf{f} \in \mathbb{R}_+^N$. We use i to index into the degraded image $\mathbf{g}$, i.e. $g_i = g(\mathbf{x}_i)$, where $\mathbf{x}_i$ is the coordinate of the i^{th} pixel. Likewise, we use j to index into the sharp image $\mathbf{f}$, such that $f_j = f(\mathbf{x}'_j)$ for a coordinate $\mathbf{x}'_j$. Finally, we note that to evaluate an image at arbitrary (sub-pixel) locations, we interpolate from nearby pixels. In this work, we use bilinear interpolation, whereby sub-pixel values of an image, say $g(\mathbf{x})$ are interpolated as a linear combination of the four nearest pixels.

[1] `http://epaperpress.com/ptlens/`

In this discrete setting, we write the blurred image as a linear function of the sharp image:

$$g_i^* = \sum_j A_{ij} f_j, \tag{1.10}$$

or in matrix-vector notation, $$\mathbf{g}^* = \mathbf{A}\mathbf{f}, \tag{1.11}$$

where the $N \times N$ matrix $\mathbf{A}$ captures the discrete point spread function (PSF). Each row of the matrix $\mathbf{A}$ corresponds to a single blurred pixel in $\mathbf{g}$, and captures the fact that each blurred pixel is a weighted sum of multiple sharp pixels. In most cases of blur, the light received by each pixel in $\mathbf{g}$ comes from a relatively small number of nearby pixels in $\mathbf{f}$. As a result, the PSF matrix $\mathbf{A}$ for an image is usually sparse (contains a relatively small number of non-zero values).

We discretise the camera orientation space into a 3D volumetric grid of size $N_X \times N_Y \times N_Z$, and assign each orientation $\boldsymbol{\theta}^{(k)}$ a weight w_k, for $k \in \{1, \ldots, K\}$, where $K = N_X N_Y N_Z$. The set of weights $\mathbf{w}$ forms a global descriptor for the camera shake blur in an image, and by analogy with convolutional blur, we refer to $\mathbf{w}$ as the *blur kernel*.

Figure 1.1 (c) shows a visualisation of $\mathbf{w}$, where the cuboidal volume of size $N_X \times N_Y \times N_Z$ is shown, with the points inside representing the non-zero elements of $\mathbf{w}$ in 3D. The kernel has also been projected onto the three back faces of the cuboid to aid visualisation, with white corresponding to a large value, and black corresponding to zero.

Each element w_k corresponds to a camera orientation $\boldsymbol{\theta}^{(k)}$, and consequently to a homography $\mathbf{H}_k$, so that in the discrete setting, the blurred image $\mathbf{g}^*$ is modelled as a weighted sum of a set of projectively-transformed versions of $\mathbf{f}$:

$$\mathbf{g}^* = \sum_k w_k \mathbf{T}^{(k)} \mathbf{f}, \tag{1.12}$$

where $\mathbf{T}^{(k)}$ is the $N \times N$ matrix which applies homography $\mathbf{H}_k$ to the sharp image $\mathbf{f}$. The matrix $\mathbf{T}^{(k)}$ is very sparse. For example, if bilinear interpolation is used when transforming the image, each row has only four non-zero elements corresponding to the interpolation weights for each pixel. By writing out Equation (1.12) for a single pixel, we obtain the discrete analog of Equation (1.7):

$$g_i^* = \sum_k w_k \left(\sum_j T_{ij}^{(k)} f_j \right), \tag{1.13}$$

where i and j index the pixels of the blurred image and the sharp image, respectively. For a blurred pixel g_i^* with coordinate vector $\mathbf{x}_i$, the sum $\sum_j T_{ij}^{(k)} f_j$ interpolates the value of the sub-pixel location $f(\mathbf{H}_k \mathbf{x}_i)$. We will return to how $\mathbf{H}_k$, and the corresponding matrices $\mathbf{T}^{(k)}$, are sampled in Section 1.7.

Due to the bilinear form of Equation (1.13), note that when either the blur kernel or the sharp image is known, the blurred image is linear in the remaining

unknowns, i.e.

$$\text{given } \mathbf{w}, \qquad \mathbf{g}^* = \mathbf{A}\mathbf{f}, \qquad \text{where } A_{ij} = \sum_k T_{ij}^{(k)} w_k, \tag{1.14}$$

$$\text{given } \mathbf{f}, \qquad \mathbf{g}^* = \mathbf{B}\mathbf{w}, \qquad \text{where } B_{ik} = \sum_j T_{ij}^{(k)} f_j. \tag{1.15}$$

In the first form, $\mathbf{A} \in \mathbb{R}^{N \times N}$ is a large sparse matrix, whose rows each contain a local blur filter acting on $\mathbf{f}$ to generate a blurred pixel. In the second form, when the sharp image is known, each column of $\mathbf{B} \in \mathbb{R}^{N \times K}$ contains a projectively transformed copy of the sharp image. We will use each of these forms in the following sections.

In contrast to this model, when the PSF is spatially-invariant, we denote the discrete convolution kernel by $\mathbf{a}$, and write

$$\mathbf{g}^* = \mathbf{a} * \mathbf{f}. \tag{1.16}$$

1.4 Blind estimation of blur from a single image

In this section, we demonstrate the effectiveness of the spatially-variant blur model presented in Section 1.3 by using it for blind deblurring of images acquired under camera shake. We consider single-image deblurring, where only a blurred image is available, and use our model to replace the spatially-invariant blur model in an existing algorithm for blind deblurring; specifically, applying our model within the algorithm proposed by Cho & Lee (2009).

"Blind" estimation of PSFs directly from images has a long history (Gull & Skilling 1984, Ayers & Dainty 1988, Fish, Brinicombe, Pike & Walker 1995), however the particular case of camera-shake blur has attracted significant attention recently (Fergus et al. 2006, Shan et al. 2008, Cai, Ji, Liu & Shen 2009, Cho & Lee 2009, Levin et al. 2009, Xu & Jia 2010, Gupta et al. 2010, Harmeling, Hirsch & Schölkopf 2010, Krishnan, Tay & Fergus 2011).

The method of Cho & Lee (2009) is similar in spirit to many other MAP-type algorithms recently proposed (Shan et al. 2008, Cai et al. 2009, Xu & Jia 2010, Krishnan et al. 2011), and proceeds by alternately updating the blur kernel and latent image, in a multi-scale framework. The algorithm iterates over three main steps. In the first step, a set of non-linear filters are applied to the current estimate $\hat{\mathbf{f}}$ of the sharp image, in order to predict the locations and magnitudes of step edges in the sharp image. This is done using a bilateral filter (Tomasi & Manduchi 1998) to denoise the image, followed by a shock filter (Osher & Rudin 1990) to enhance sharp edges, producing an image $\hat{\mathbf{f}}'$. Finally, the derivatives of $\hat{\mathbf{f}}'$ are computed, and thresholded based on their orientation and magnitude, to produce a set of sparse edge maps $\{\mathbf{p}^{(\mathbf{q})}\}$.

In the second step, these predicted edges are used to estimate the blur kernel by solving a linear least-squares problem. Having found the kernel $\hat{\mathbf{a}}$ that minimises

this problem, any elements whose value are below a threshold are set to zero. This encourages sparsity in the kernel, and ensures that all the elements are positive. In the third step the sharp image is estimated, using the current estimate of the blur kernel $\hat{\mathbf{a}}$ to deconvolve the blurred image and obtain an improved estimate of the sharp image $\hat{\mathbf{f}}$. This step is also performed by solving a linear least-squares problem.

These three steps are applied iteratively, working from coarse to fine in a multi-scale framework. The iterative process generally converges quickly at each scale, and 5-7 iterations are typically sufficient. The kernel-update and image-update sub-problems involved in the algorithm of Cho & Lee (2009) are linear least-squares by virtue of the fact that convolution is a bilinear operation on the sharp image and the blur kernel. The fact that our blur model, given in Equation (1.13) is bilinear in the sharp image and blur kernel is the key feature that allows it to be applied within this, and other deblurring algorithms.

Although we have chosen the algorithm of Cho & Lee (2009) to apply our blur model, we note that many MAP-style blind deblurring algorithms consist of the same basic blocks, alternating between updating the kernel and the latent image by solving linear least-squares sub-problems, often in a multi-scale framework (Shan et al. 2008, Xu & Jia 2010, Krishnan et al. 2011). As such, the following discussion can apply equally well to a number of algorithms.

1.4.1 Updating the blur kernel

The general form of the spatially-invariant kernel update problem (the second step) in the algorithm of Cho & Lee (2009) is

$$\min_{\mathbf{a}} \ \left\|\mathbf{a} * \mathbf{f} - \mathbf{g}\right\|_2^2 + \beta \left\|\mathbf{a}\right\|_2^2, \tag{1.17}$$

where the first data reconstruction term measures how well the blur kernel $\mathbf{a}$ applied to the current estimate of the latent image $\mathbf{f}$ reconstructs the observed blurred image $\mathbf{g}$, and the second term, weighted by a scalar parameter β, is the ℓ_2 regularisation of the blur kernel $\mathbf{a}$. Note that in Cho & Lee's (2009) formulation, the data term replaces $\mathbf{f}$ with the sparse edge maps $\mathbf{p}^{(q)}$ produced by the non-linear filtering of the current estimate of the latent image $\hat{\mathbf{f}}$, i.e. has the form $\left\|\mathbf{a} * \mathbf{p}^{(q)} - \mathbf{d}^{(q)} * \mathbf{g}\right\|_2^2$, where q indexes over the first and second-order spatial derivatives, and $\mathbf{d}^{(q)}$ are spatial derivative filters.

When applying our spatially-variant blur model, we modify the problem given by Equation (1.17) to update the spatially variant kernel $\mathbf{w}$ as

$$\min_{\mathbf{w}} \ \left\|\mathbf{B}\mathbf{w} - \mathbf{g}\right\|_2^2 + \beta \sum_k w_k, \quad \text{s.t.} \ \ \forall k \ \ w_k \geq 0, \tag{1.18}$$

where $\mathbf{B}$ is given by Equation (1.15). Here we have substituted our blur model into the data-reconstruction term, and replaced the ℓ_2 Tykhonov regularisation with ℓ_1 regularisation and non-negativity constraints on the kernel. Now,

instead of simply a linear least-squares problem to update $\mathbf{w}$, we have an instance of the *Lasso* problem (Tibshirani 1996), for which efficient optimisation algorithms exist (Efron, Hastie, Johnstone & Tibshirani 2004, Kim, Koh, Lustig, Boyd & Gorinevsky 2007, Mairal, Bach, Ponce & Sapiro 2010). Similarly to (Cho & Lee 2009) we use sparse edge maps $\mathbf{p}^{(q)}$ which results, after expanding $\mathbf{B}$ using Equation (1.15), in a data term of the form $\left\|\sum_k w_k \mathbf{T}^{(k)} \mathbf{p}^{(q)} - \mathbf{d}^{(q)} * \mathbf{g}\right\|_2^2$.

The need for the ℓ_1 (instead of the simpler ℓ_2) regularisation arises from the differences between our kernel and convolution kernels. Fundamentally, our kernels cover a larger space of image transformations than convolution kernels (3-parameter homographies instead of 2D image translations), and we must estimate more parameters from the same amount of data. As a result, the PSF estimation process is liable to become poorly conditioned, due to an increased amount of ambiguity in the data-reconstruction term. We have observed that when simply replacing the convolution in Equation (1.17) with our model, but without changing the regularisation, the resulting 3D kernels contain many non-zeros spread smoothly throughout, and do not produce good deblurred outputs (see Figure 1.5). On the other hand, with ℓ_1 regularisation, the optimisation is more likely to choose between ambiguous camera orientations than spreading non-zero values across all of them. In the remainder of the chapter, we refer to the original algorithm of Cho & Lee as MAP-ℓ_2, and our ℓ_1 regularised version as MAP-ℓ_1.

In addition to the use of ℓ_1 regularisation, we note that in order to constrain the 3D kernel adequately, we require data from all regions of the image. This can be seen by considering a vertical blur at the left or right-hand side of the image. Such a blur could be explained either by a rotation of the camera about its X axis, or a rotation about its Z axis. In order to resolve this ambiguity, we would need to look at other regions of the image. To ensure that we use observations from all parts of the image when updating the kernel, we modify the construction of the edge-maps $\mathbf{p}^{(q)}$ from the filtered sharp image $\hat{\mathbf{f}}'$. We simply subdivide the image into 3×3 regions, and apply the gradient thresholding step independently on each.

1.4.2 Updating the latent image

The latent image update (the third step) in the algorithm of Cho & Lee (2009) is performed in a similar way to many non-blind deblurring algorithms (Xu & Jia 2010, Krishnan & Fergus 2009) by solving a linear least-squares problem of the form

$$\min_{\mathbf{f}} \ \|\mathbf{a} * \mathbf{f} - \mathbf{g}\|_2^2 + \alpha \|\mathbf{d}^x * \mathbf{f}\|_2^2 + \alpha \|\mathbf{d}^y * \mathbf{f}\|_2^2 , \tag{1.19}$$

where $\mathbf{d}^x$ and $\mathbf{d}^y$ represent horizontal and vertical derivative filters, and α is a regularisation weight. Note that in Cho & Lee's (2009) formulation, the data term also takes into account the derivatives of $\mathbf{f}$ and $\mathbf{g}$ as well as the intensities (i.e. $\left\|\mathbf{a} * (\mathbf{d}^{(q)} * \mathbf{f}) - \mathbf{d}^{(q)} * \mathbf{g}\right\|_2^2$ is also penalised, for various derivative filters $\mathbf{d}^{(q)}$).

To apply our blur model, we simply replace the convolution in the data reconstruction term and update the latent image as

$$\min_{\mathbf{f}} \left\|\mathbf{A}\mathbf{f} - \mathbf{g}\right\|_2^2 + \alpha \left\|\mathbf{d}^x * \mathbf{f}\right\|_2^2 + \alpha \left\|\mathbf{d}^y * \mathbf{f}\right\|_2^2, \tag{1.20}$$

where $\mathbf{A}$ is given by Equation (1.14). We minimise Equation (1.20) using conjugate-gradient descent (Shewchuk 1994).

Note that at this point, we are not able to take full advantage of the speed optimisations proposed by Cho & Lee (2009), due to their use of Fourier transforms to compute convolutions and to perform direct minimisation of Equation (1.19) in the frequency domain. However, in the following section we will describe an efficient approximation of the spatially-variant blur model which enables this.

1.5 Efficient computation of the spatially-variant model

Due to the additional computational expense incurred by using a spatially-variant blur model instead of a spatially-invariant one, both blind and non-blind deblurring under this model can be very time consuming. As seen in the previous section, MAP-type deblurring algorithms typically involve solving linear least-squares problems, minimising $\|\mathbf{A}\mathbf{f} - \mathbf{g}\|_2^2$ with respect to $\mathbf{f}$, and $\|\mathbf{B}\mathbf{w} - \mathbf{g}\|_2^2$ with respect to to $\mathbf{w}$. As defined in Equations (1.14) and (1.15), the matrices $\mathbf{A}$ and $\mathbf{B}$ involve sums over a set of homographies $\{\mathbf{T}^{(\mathbf{k})}\}$, where the size K of the set can be large, on the order of hundreds or thousands.

In iterative minimisation algorithms for solving such least-squares problems, we must repeatedly compute arbitrary matrix-vector multiplications involving $\mathbf{A}$ and $\mathbf{B}$. For images with millions of pixels these matrices are generally too large to fit into memory, and the matrix-vector products must be computed on-the-fly, by explicitly warping the image for each homography $\mathbf{T}^{(k)}$. This is by far the biggest bottleneck in the use of our model, and means that both blind PSF estimation and non-blind deblurring are significantly slower than for spatially-invariant blur.

To reduce the running time of the whole deblurring process, in this section we propose an efficient approximation to the blur model from Equation (1.12), based on the locally-uniform "Efficient Filter Flow" proposed by Hirsch, Sra, Schölkopf & Harmeling (2010). We begin by describing the approximation in Section 1.5.1, before demonstrating in Section 1.5.2 how it can be used to compute the matrix-vector products necessary for the kernel update step (Equation (1.18)) very quickly. In Section 1.5.3 we describe how this approach allows us to update the sharp image (Equation (1.20)) directly in the frequency domain (instead of using iterative methods). The approximation allows blind deblurring to be performed an order of magnitude faster than when using the exact forward model. Note that concurrently with our work, Hirsch, Schuler, Harmeling & Schölkopf (2011) proposed a similar model for efficiently computing spatially-variant camera-shake blur.

1.5.1 A locally-uniform approximation for camera shake

Hirsch et al. (2010) observe that in some cases of spatially-variant image blur, the blur may vary slowly and smoothly across the image. In these cases, it is reasonable to approximate spatially-variant blur as locally-uniform. Following this observation, they propose a model for spatially-variant blur, whereby the sharp image $\mathbf{f}$ is covered with a coarse grid of P overlapping patches, each of which is modelled as having a spatially-invariant blur. The overlapping patches ensure the blur varies smoothly, while allowing the forward model to be computed using P small convolutions. Hirsch et al. (2010) assign each patch r a spatially-invariant blur filter $\mathbf{a}^{(r)}$, and their model is given by:

$$\mathbf{g}^* = \sum_{r=1}^{P} {\mathbf{C}^{(r)}}^\top \left(\mathbf{a}^{(r)} * \left(\mathbf{m} \circ \mathbf{C}^{(r)} \mathbf{f} \right) \right), \tag{1.21}$$

where $\mathbf{C}^{(r)}$ is a matrix that crops the r^{th} patch of the image $\mathbf{f}$ (thus ${\mathbf{C}^{(r)}}^\top$ reinserts it via zero-padding). The vector $\mathbf{m}$ is the Bartlett-Hann window, and $\cdot \circ \cdot$ represents the Hadamard (element-wise) product. Note that this model can be computed very efficiently by computing the discrete Fourier transforms of each patch and filter using the fast Fourier transform (FFT), multiplying them element-wise in the frequency domain, and then taking the inverse discrete Fourier transform of the result. Under varying assumptions, different authors have also proposed locally-uniform models of spatially-variant blur, which take similar forms to Equation (1.21) (Nagy & O'Leary 1998, Vio, Nagy, Tenorio & Wamsteker 2005, Tai, Du, Brown & Lin 2010).

In their original work, Hirsch et al. (2010) parameterise the blur using a separate filter $\mathbf{a}^{(r)}$ for each patch r. Likewise Harmeling et al. (2010), who apply this model to single-image camera shake removal, also estimate a separate filter per patch using the MAP algorithm of Cho & Lee (2009). One weakness of this approach is that in textureless regions, the algorithm of Cho & Lee may fail, and so heuristics are needed to encourage similarity between neighbouring filters, and also to detect and replace failed local kernel estimates.

Given the forward blur model for camera shake in Equation (1.12), which is parameterised by a single set of weights $\mathbf{w}$, we can in fact write each $\mathbf{a}^{(r)}$ in terms of $\mathbf{w}$. For each patch r, we choose $\mathbf{a}^{(r)}$ to be the point spread function for the central pixel i_r, which is given by the i_r^{th} row of $\mathbf{A}$. Since $\mathbf{A}$ is linear in $\mathbf{w}$, we can construct a matrix $\mathbf{J}^{(r)}$ such that $\mathbf{a}^{(r)} = \mathbf{C}^{(r)} \mathbf{J}^{(r)} \mathbf{w}$. The elements of each $\mathbf{J}^{(r)}$ are simply a re-arrangement of the elements of the matrices $\mathbf{T}^{(k)}$; $J^{(r)}_{jk} = T^{(k)}_{i_r j}$.

Having written each filter $\mathbf{a}^{(r)}$ in terms of $\mathbf{w}$, we can then substitute this into Equation (1.21) to obtain the following approximation of the forward model from Equation (1.12):

$$\mathbf{g}^* = \mathbf{A}\mathbf{f} = \mathbf{B}\mathbf{w} \simeq \sum_{r=1}^{P} {\mathbf{C}^{(r)}}^\top \left(\left(\mathbf{C}^{(r)} \mathbf{J}^{(r)} \mathbf{w} \right) * \left(\mathbf{m} \circ \mathbf{C}^{(r)} \mathbf{f} \right) \right). \tag{1.22}$$

This equation allows the forward model to be computed quickly using only a handful of small convolutions, which can be performed efficiently in the frequency domain. Figure 1.4 shows how the quality of the locally uniform approximation varies with the number of patches being used, compared to the exact model. In all our experiments, we use a grid of 6×8 patches.

1.5.2 Updating the blur kernel

In iterative algorithms for estimating the blur kernel $\mathbf{w}$ from Equation (1.18), we typically need to compute the gradient of $\|\mathbf{B}\mathbf{w} - \mathbf{g}\|_2^2$ with respect to $\mathbf{w}$, which involves computing $\mathbf{B}^\top\mathbf{y}$ for arbitrary $\mathbf{y}$ and $\mathbf{B}^\top\mathbf{B}$. In addition to being able to compute the forward model quickly, Equation (1.22) provides us with a fast approximate way of computing these products using one convolution per patch:

$$\mathbf{B}^\top\mathbf{y} \simeq \sum_{r=1}^{P} {\mathbf{J}^{(r)}}^\top {\mathbf{C}^{(r)}}^\top \Big(\big(\mathbf{m} \circ \mathbf{C}^{(r)}\mathbf{f}\big) \otimes \big(\mathbf{C}^{(r)}\mathbf{y}\big) \Big) \tag{1.23}$$

$$\mathbf{B}^\top\mathbf{B} \simeq \sum_{r=1}^{P} {\mathbf{J}^{(r)}}^\top {\mathbf{C}^{(r)}}^\top \operatorname{XCorrMatrix}\Big(\big(\mathbf{m} \circ \mathbf{C}^{(r)}\mathbf{f}\big) \otimes \big(\mathbf{C}^{(r)}\mathbf{f}\big) \Big) \mathbf{C}^{(r)}\mathbf{J}^{(r)}, \tag{1.24}$$

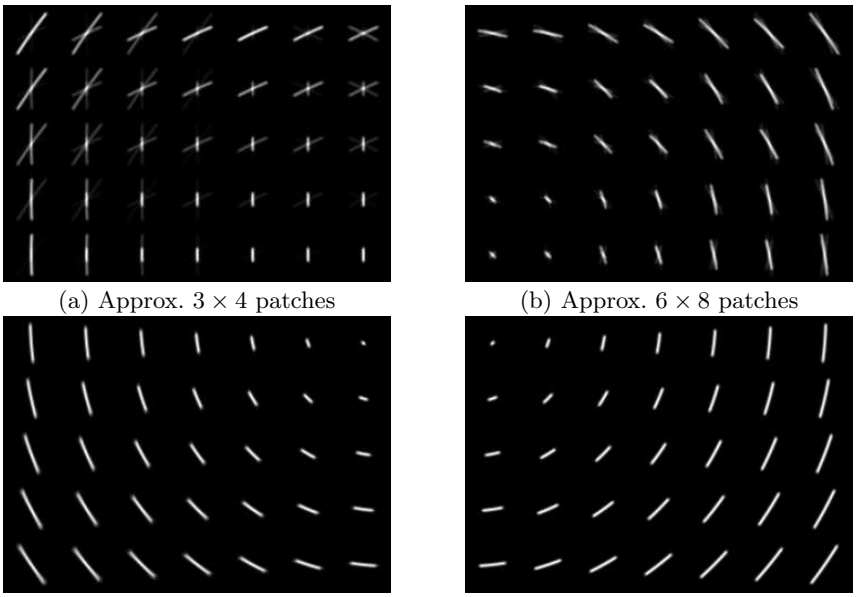
where $\cdot \otimes \cdot$ represents 2D correlation, and the function XCorrMatrix constructs the full cross-correlation *matrix* from a cross-correlation *vector* between two signals by replicating and shifting elements. A cross-correlation matrix $\mathbf{M}$ for a pair of signals $\mathbf{u}$ and $\mathbf{v}$ stores their inner-product at all possible translations of both signals. Assuming appropriate boundary conditions, each row of $\mathbf{M}$ is simply a shifted version of the cross-correlation $\mathbf{u} \otimes \mathbf{v}$.

1.5.3 Updating the latent image: fast, non-iterative non-blind deconvolution

The locally-uniform approximation allows the forward model and its derivatives to be computed much faster using the FFT, such that we can quickly compute the $\mathbf{A}^\top\mathbf{y}$ and $\mathbf{A}^\top\mathbf{A}$ products needed to perform gradient-based optimisation of the sharp image in Equation (1.20). However, for spatially-invariant blur, the fastest way of updating the sharp image (in Equation (1.19)) is not using iterative methods such as conjugate-gradient descent, but rather using non-iterative frequency-domain deconvolution, which we outline below. In this section we extend this method to handle spatially-variant blur, via the locally-uniform approximation.

When blur is spatially-invariant, using Parseval's theorem, Equation (1.19) can be transformed into N independent 1D quadratic minimisations in the frequency domain, allowing the solution to be obtained directly by pixel-wise division in the frequency domain (Gamelin 2001):

$$\hat{\mathbf{f}} = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(\mathbf{a})^* \circ \mathcal{F}(\mathbf{g})}{\mathcal{F}(\mathbf{a})^* \circ \mathcal{F}(\mathbf{a}) + \alpha\left(\mathcal{F}(\mathbf{d}^x)^* \circ \mathcal{F}(\mathbf{d}^x) + \mathcal{F}(\mathbf{d}^y)^* \circ \mathcal{F}(\mathbf{d}^y)\right)} \right), \tag{1.25}$$



(a) Approx. 3×4 patches (b) Approx. 6×8 patches

(c) Approx. 12×16 patches (d) Exact model

Figure 1.4 Approximating spatially-variant blur by combining uniformly-blurred, overlapping patches. Using the model described in Section 1.5.1, we can efficiently compute approximations to the spatially-variant blur model in Equation (1.12). With a small number of patches (a), the PSF at each pixel is visibly the sum of different blurs from overlapping patches. As more patches are used (b–c), the approximation becomes increasingly close to the exact model (d) – at 12×16 patches it is almost indistinguishable.

where $\mathcal{F}(\,\cdot\,)$ takes the 2D discrete Fourier transform (computed using the fast Fourier transform), and $\mathcal{F}^{-1}(\,\cdot\,)$ the inverse Fourier transform.

However, for spatially-variant blur, the locally-uniform approximation does not immediately permit this. Even though each patch has a spatially-invariant blur, the fact that the patches overlap means that the reconstruction of any blurred pixel will involve several patches. This can be seen by inserting the locally-uniform model into the non-blind deblurring problem in Equation (1.20), and seeing that the sum over patches lies inside the data-reconstruction term:

$$\min_{\mathbf{f}} \; \Big\| \sum_{r=1}^{P} \mathbf{C}^{(r)\top} \big(\mathbf{a}^{(r)} * (\mathbf{m} \circ \mathbf{C}^{(r)} \mathbf{f}) \big) - \mathbf{g} \Big\|_2^2 + \alpha \big\| \mathbf{d}^x * \mathbf{f} \big\|_2^2 + \alpha \big\| \mathbf{d}^y * \mathbf{f} \big\|_2^2. \quad (1.26)$$

Thus, this equation cannot be minimised independently for each patch.

In order to be able to estimate each patch independently, one simple solution is to use Jensen's inequality to obtain an upper bound on Equation (1.26), taking the sum over patches outside the data-reconstruction term, and expanding the regularisation terms to also penalise individual patches. Having done this, we minimise the upper bound instead of the original cost. We define the set of

blurred patches $\{\mathbf{g}^{(\mathbf{r})}\}$ and sharp patches $\{\mathbf{f}^{(\mathbf{r})}\}$ such that $\mathbf{g}^{(r)} = \mathbf{m} \circ \mathbf{C}^{(r)}\mathbf{g}$ and $\mathbf{f}^{(r)} = \mathbf{m} \circ \mathbf{C}^{(r)}\mathbf{f}$, and estimate each deblurred patch $\hat{\mathbf{f}}^{(r)}$ by solving

$$\min_{\mathbf{f}^{(r)}} \left\|\mathbf{a}^{(r)} * \mathbf{f}^{(r)} - \mathbf{g}^{(r)}\right\|_2^2 + \alpha\left\|\mathbf{d}^x * \mathbf{f}^{(r)}\right\|_2^2 + \alpha\left\|\mathbf{d}^y * \mathbf{f}^{(r)}\right\|_2^2 \tag{1.27}$$

for each patch. This can be done using the direct frequency-domain method of Equation (1.25). Second, we estimate the full deblurred image $\hat{\mathbf{f}}$ that best matches the deblurred patches $\{\hat{\mathbf{f}}^{(r)}\}$ by minimising $\sum_{r=1}^{P}\left\|\mathbf{m} \circ \mathbf{C}^{(r)}\mathbf{f} - \hat{\mathbf{f}}^{(r)}\right\|_2^2$. This problem can be solved independently for each pixel, yielding the following solution:

$$\hat{\mathbf{f}} = \frac{\sum_r {\mathbf{C}^{(r)}}^\top(\mathbf{m} \circ \hat{\mathbf{f}}^{(r)})}{\sum_r {\mathbf{C}^{(r)}}^\top(\mathbf{m} \circ \mathbf{m})}. \tag{1.28}$$

In Figure 1.7 we compare this method of non-blind deblurring to other possibilities for solving Equation (1.20) with spatially-variant camera shake blur. The fast independent method produces results which are visually very similar to those obtained using the exact model, in a significantly shorter amount of time. Using this direct deconvolution method to update the latent image via Equation (1.20) during blind deblurring provides an additional speed improvement, compared to the use of conjugate-gradient descent with the approximate forward model.

As a demonstration of the speed-up achievable with this approximation, the examples in Figures 1.5 to 1.7 all took more than 3 hours for blind deblurring using the exact model (implemented in MATLAB and C), compared to under 6 minutes using the approximation presented in this section (implemented in MATLAB). The results shown in this chapter were all produced using the approximation. The reduction in computational complexity can be quantified by comparing the exact model in Equation (1.13) with the approximate model in Equation (1.22). Evaluating Equation (1.13) requires $\mathcal{O}(NK)$ operations, where N is the number of pixels in the image and K is the number of non-zeros in $\mathbf{w}$, whereas Equation (1.22) requires $\mathcal{O}(N \log N)$ operations (due to the use of the FFT to compute the convolution). For a camera shake blur, $\mathcal{O}(K) \geq \mathcal{O}(\sqrt{N})$, and thus the approximation provides a significant reduction in computational complexity. For a complete discussion of the computational complexity, as well as a full derivation of the equations presented here, please refer to Whyte (2012).

Application to other non-blind deblurring algorithms

The fast, non-iterative method for non-blind deblurring presented in this section can be used as a building-block in more sophisticated non-blind deblurring algorithms. A number of recent algorithms, which place sparse-gradient priors on the deblurred image (Krishnan & Fergus 2009, Wang, Yang, Yin & Zhang 2008, Shan et al. 2008), or which use more robust data-reconstruction terms (Yan, Zhang & Yin 2009, Xu & Jia 2010), involve solving sub-problems of the form of Equation (1.19), which has a quadratic data-reconstruction term and quadratic regularisation. As such, the non-iterative method presented in this section can be

used to extend these more sophisticated algorithms to handle spatially-variant blur, without a substantial increase in processing time. The results shown in Section 1.6 use the algorithm of Krishnan & Fergus (2009), modified in this way, to perform the final non-blind deblurring.

1.6 Single-image deblurring results

In this section we present results of single-image deblurring using the MAP-ℓ_1 algorithm to estimate the spatially-variant PSF, with comparisons to results obtained with the original algorithm of Cho & Lee (2009) on real data. Having estimated the PSF, we apply the non-blind deblurring algorithm of Krishnan & Fergus (2009) to estimate the final deblurred image. This algorithm is easily adapted to non-uniform blur since it involves repeated minimisations of quadratic cost functions similar to Equation (1.20).

Figure 1.5 shows a blind deblurring result on an image blurred by real camera shake. Our model is able to model and remove the blur, while the results with the original algorithm contain visible artefacts. This is explained by both the wide field of view, and the fact that the kernels estimated using our algorithm exhibit significant in-plane rotation. Also shown is the result of using ℓ_2 regularisation on the spatially-variant kernel. As discussed in Section 1.4.1, the kernel produced is not sparse, and as a result the deconvolved output exhibits many artefacts compared to the MAP-ℓ_1 result.

Figure 1.6 shows another example of single-image deblurring, using the MAP algorithm. While the uniform blur kernel provides a reasonable estimate of the true blur, and allows us to resolve some of the text on the book's cover, the use of our non-uniform blur model provides a clear improvement, and permits almost all of the text to be read.

Additional blind deblurring results of real camera shakes are shown in figures Figure 1.1 and Figure 1.7.

1.6.1 Limitations and failures

Since the MAP approach to blind deblurring attempts to solve a non-convex minimisation problem, it is not possible to guarantee a globally optimal solution. However, in practice we have found the MAP-ℓ_1 algorithm to be capable of deblurring a wide range of images with large blurs – the blurs removed in this chapter are up to 35 pixels wide (e.g. Figure 1.6) – for both uniform and non-uniform blur. For our model, this corresponds to around 3° – 5° of rotation around each axis for a photograph whose width and focal length are both 1000 pixels. This is due in large part to the multi-scale approach; by finding a sequence of solutions at increasingly fine resolutions, the large scale structures in the blur kernel and sharp image are resolved before the fine details.

Nevertheless, failures do occur and Figure 1.8 shows an example case. We

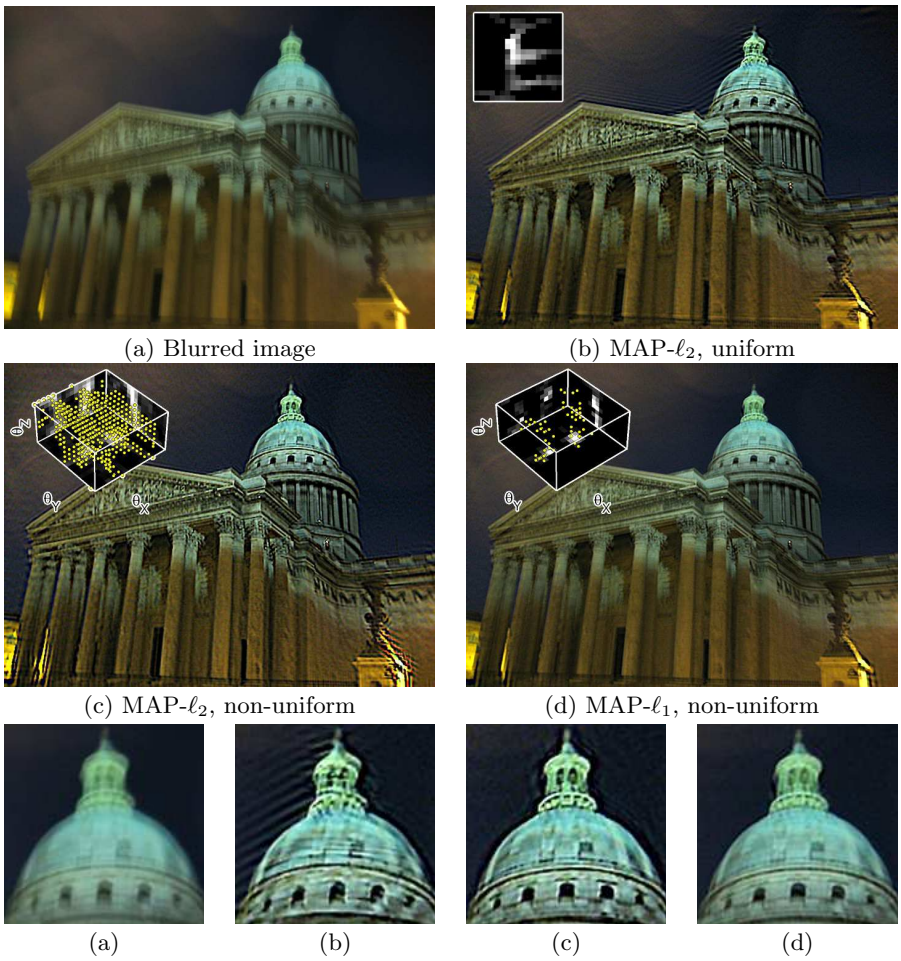


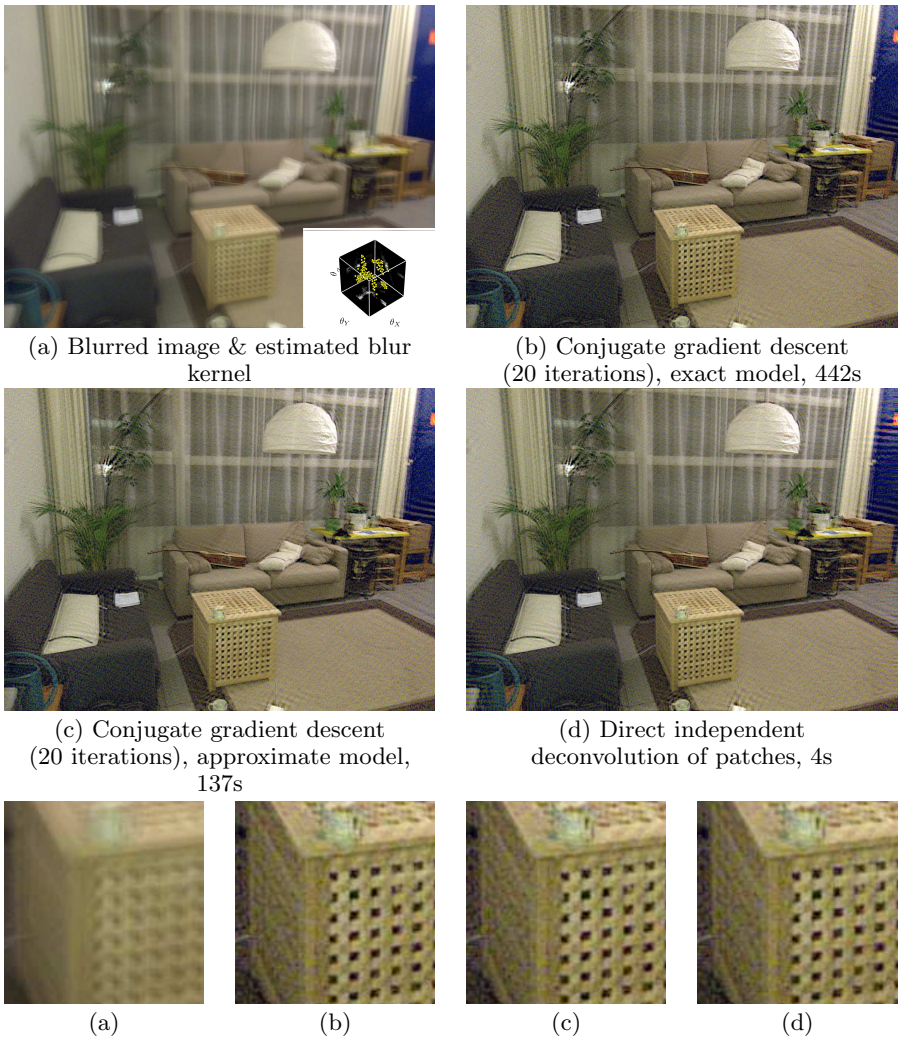
Figure 1.5 Blind deblurring of real camera shake. A hand-held image with camera shake (a), captured with a shutter speed of 1 second, with the results of blind deblurring using the algorithm of Cho & Lee with (b) a uniform blur model and (c–d) our spatially-varying blur model. The estimated kernels are shown inset in the deblurred results. The result using our blur model in the MAP-ℓ_1 algorithm (d) shows more detail and fewer artefacts than those using the uniform blur model, as can be seen in the zoomed-in portions shown in the last row. Also shown is the result when using our blur model with ℓ_2 regularisation on the kernel (c). As can be seen, the ℓ_2 regularisation is not sufficient to produce a good estimate of the kernel, and results in a deblurred output containing many artefacts. The rotational blur kernels in (c–d) cover $\pm 0.7^\circ$ in θ_X and θ_Y and $\pm 1.4^\circ$ in θ_Z.

have observed failures caused by several factors, including a high level of noise in the input images, an excessively large blur, or an unknown non-linear camera response function e.g. Figure 1.8 (bottom row). Also, the algorithm may fail when the edge-based heuristics which guide the blind deblurring do not match

(a) Blurred image (b) Some local PSFs for (d), magnified

(c) MAP-ℓ_2, uniform (d) MAP-ℓ_1, non-uniform

Figure 1.6 Blind deblurring of real camera shake. The result of blind deblurring on a real camera shake image (a), captured with a shutter speed of 1 second, using the MAP approach of Cho & Lee with both the uniform and non-uniform blur models. Also shown in (b) are some of the local PSFs generated from the blur kernel in (d) at various points in the image. In the blurred image, most of the text on the book cover is too blurred to read. Deblurring the image with the uniform blur model (c) allows some of the text on the cover of the book to be read, however, after deblurring with our non-uniform model (d), all but the smallest text becomes legible. The estimated kernels for the two models are shown inset in the deblurred results. The blur kernel in (d) covers $\pm 0.4^\circ$ in θ_X and θ_Y, and $\pm 0.9^\circ$ in θ_Z. Reproduced from (Whyte et al. 2012) with permission, © Springer 2012.



(a) Blurred image & estimated blur kernel

(b) Conjugate gradient descent (20 iterations), exact model, 442s

(c) Conjugate gradient descent (20 iterations), approximate model, 137s

(d) Direct independent deconvolution of patches, 4s

(a) (b) (c) (d)

Figure 1.7 Least-squares non-blind deblurring using the exact and approximate forward models. Given a blurred image of size 1024×768 and blur kernel (a), this figure shows the results and computation times for least-squares deconvolution with ℓ_2 gradient regularisation, using (b) conjugate-gradient descent (CG) with the exact forward model, (c) CG with the approximate forward model, and (d) direct deconvolution using the approach described in Section 1.5.3. The results are visually similar using all three methods. Using CG with the approximate forward model is much faster than with the exact model, however the direct approach takes only a fraction of the time of either of these.

the particular image being estimated, e.g. in an image that contains only fine-scale texture with no large-scale step edges.

Since we have assumed that camera translation has a negligible blurring effect,

(a) Blurred image (b) Deblurred result from MAP-ℓ_1 algorithm (c) Kernel estimated by MAP-ℓ_1 algorithm

Figure 1.8 Blind deblurring failures. A blurred image (a) with the deblurred result (b) and kernel (c) estimated by the MAP-ℓ_1 algorithm. This image contains a large amount of noise, and was captured with a camera-phone whose response function is unknown. As a result, the algorithm fails to estimate a good kernel, and instead returns a kernel which is close to a delta function.

our model (and the uniform model too) is unlikely to produce good results on images for which this is not true, due to the depth-dependent blur produced. As discussed in Section 1.2, this is unlikely to be a problem on most shaken images, except for close-up photos where the subject is less than about 1 m from the camera.

1.7 Implementation

The implementation of the algorithm of Cho & Lee (2009) is our own, and we use this implementation for both uniform and non-uniform blur models when comparing results. A binary executable for Cho & Lee's (2009) algorithm is available, however we did not observe an improvement in the results obtained, and thus use our own implementation to permit a fairer comparison between the results from the uniform and non-uniform blur models. The implementation of the non-blind deblurring algorithm of Krishnan & Fergus (2009) is based on MATLAB code made available online by the authors[2].

Sampling the set of rotations

One important detail to consider is how finely to discretise the orientation parameter $\boldsymbol{\theta}$. In the discrete case, each grid point $\boldsymbol{\theta}^{(k)}$ corresponds to a transformation matrix $\mathbf{T}^{(k)}$ in the sum in Equation (1.12). Undersampling the space of orientations will affect our ability to accurately reconstruct the blurred image, but sampling it too finely will lead to unnecessary computation. Since the kernel is defined over the 3 parameters θ_X, θ_Y and θ_Z, doubling the sampling resolution increases the number of kernel elements by a factor of 8. In practice, we have found that a good choice of grid spacing is that which corresponds to a maximum displacement of 1 pixel in the image. Since we are fundamentally limited

[2] `http://cs.nyu.edu/ dilip/research/fast-deconvolution/`

by the resolution of our images, reducing the spacing further leads to redundant orientations, which are indistinguishable from their neighbours. We set the size of the 3D kernel in terms of the size of the blur we are attempting to remove, typically a few degrees along each dimension of $\boldsymbol{\theta}$, e.g. $[-5°, 5°]$.

Multiscale implementation

Most successful blind kernel estimation algorithms here are applied within a multiscale framework, starting with a coarse representation of image and kernel, and repeatedly refining the estimated kernel at higher resolutions. In the case of single-image deblurring, this is essential to avoid poor local minima, however in our case, it is also important for computational reasons. At the original image resolution, the kernel may have thousands or tens of thousands of elements, however very few of these should have non-zero values.

Thus, in all of the applications presented in this chapter, which estimate the kernel iteratively, we use our current estimate of the kernel $\hat{\mathbf{w}}_s$ at a scale s to constrain our estimate at the next iteration. To do this, we define an "active region" where $\hat{\mathbf{w}}_s$ is non-zero, and constrain the non-zeros at the next iteration to lie within this region. By clamping many kernel elements to zero, we eliminate a large amount of computation and memory requirements associated with estimating those elements' values. We first build Gaussian pyramids for the blurred image, and at the coarsest scale $s = 0$, define the active region to cover the full kernel. At each iteration, we find the non-zero elements of our current estimate of the kernel $\hat{\mathbf{w}}_s$, and dilate this region using a $3 \times 3 \times 3$ cube to define the active region for the next iteration. When moving from one scale s to the next scale $s + 1$, we upsample $\hat{\mathbf{w}}_s$ using bilinear interpolation, find the non-zero elements of this upsampled kernel, and as before, dilate this region using a $3 \times 3 \times 3$ cube. This initialises the active region for our next estimate $\hat{\mathbf{w}}_{s+1}$. We repeat this process at each scale, until we have found the optimal kernel at the finest scale.

This approach is generally effective at reducing the computational burden of the kernel estimation without reducing accuracy, however, problems may occur if the blur kernel contains long faint structures, as it is possible for these to be clamped to zero at a coarse scale and never to be recovered.

Running time

For a 1024×768 image, our C implementation of the exact model in Equation (1.12) takes approximately 5 seconds to compute, compared to 2 seconds for our MATLAB implementation of the approximate forward model in Equation (1.22), on an Intel Xeon 2.93GHz CPU.

Our implementation, in MATLAB and C, of the MAP-ℓ_1 algorithm for spatially-variant blur takes over 3 hours to deblur a 1 megapixel (1024×768) image, depending on the size of the blur. This is a significant departure from the spatially-invariant algorithm of Cho & Lee (2009), who report deblurring times of under one minute using their C++ implementation. Using the efficient approximation

described in Section 1.5, we are able to perform blind deblurring of the same images, using our spatially-variant blur model in under 6 minutes.

1.8 Conclusion

In this chapter we have proposed a geometrically-derived model for blur caused by camera shake. For a static scene and a camera with known focal length, we have shown that the blur caused by camera rotation can be modelled using a weighted set of homographies, and have proposed a practical formulation of this model in which the blurred image is bilinear in the sharp image and the weights. We have applied our model for spatially-variant camera shake blur within an existing camera shake removal algorithms, and validated the model with experiments demonstrating superior results compared to the spatially-invariant blur model. We have also described how an efficient approximation for spatially-variant blur can be used to reduce the computational cost of computing the spatially-variant forward blur model.

Although we have demonstrated our model only in the algorithm of Cho & Lee (2009) for blind deblurring, it could equally be used with more recent algorithms, which have shown superior results, such as those of Xu & Jia (2010) and Krishnan et al. (2011). Equally, faster or more sophisticated methods for non-blind deblurring, such as those of Afonso, Bioucas-Dias & Figueiredo (2010) and Zoran & Weiss (2011), could be extended to use our spatially-variant blur model.

Interested readers can access an online demonstration of the blind deblurring algorithm at `http://www.di.ens.fr/willow/research/deblurring/`.

Acknowledgements

This work was supported in part by the MSR-INRIA laboratory, the EIT ICT labs, ERC grants VisRec and VideoWorld, and by the Institut Universitaire de France.

References

Afonso, M., Bioucas-Dias, J. & Figueiredo, M. (2010), 'Fast image recovery using variable splitting and constrained optimization', *IEEE Transactions on Image Processing* **19**(9), 2345–2356.

Ayers, G. R. & Dainty, J. C. (1988), 'Iterative blind deconvolution method and its applications', *Optics Letters* **13**(7).

Cai, J.-F., Ji, H., Liu, C. & Shen, Z. (2009), Blind motion deblurring from a single image using sparse approximation, *in* 'Proceedings of the 22nd IEEE Conference on Computer Vision and Pattern Recognition'.

Cho, S. & Lee, S. (2009), 'Fast motion deblurring', *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2009)* **28**(5), 145:1–145:8.

Efron, B., Hastie, T., Johnstone, L. & Tibshirani, R. (2004), 'Least angle regression', *Annals of Statistics* **32**(2), 407–499.

Fergus, R., Singh, B., Hertzmann, A., Roweis, S. T. & Freeman, W. T. (2006), 'Removing camera shake from a single photograph', *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2006)* **25**(3), 787–794.

Fish, D. A., Brinicombe, A. M., Pike, E. R. & Walker, J. G. (1995), 'Blind deconvolution by means of the Richardson-Lucy algorithm', *Journal of the Optical Society of America A* **12**(1), 58–65.

Gamelin, T. W. (2001), *Complex Analysis*, Springer-Verlag, New York.

Gull, S. & Skilling, J. (1984), 'Maximum entropy method in image processing', *Communications, Radar and Signal Processing, IEE Proceedings F* **131**(6), 646–659.

Gupta, A., Joshi, N., Zitnick, C. L., Cohen, M. & Curless, B. (2010), Single image deblurring using motion density functions, *in* 'Proceedings of the 11th European Conference on Computer Vision'.

Harmeling, S., Hirsch, M. & Schölkopf, B. (2010), Space-variant single-image blind deconvolution for removing camera shake, *in* 'Advances in Neural Information Processing Systems'.

Hartley, R. I. & Zisserman, A. (2004), *Multiple View Geometry in Computer Vision*, second edn, Cambridge University Press.

Hirsch, M., Schuler, C. J., Harmeling, S. & Schölkopf, B. (2011), Fast removal of non-uniform camera shake, *in* 'Proceedings of the 13th International Conference on Computer Vision'.

Hirsch, M., Sra, S., Schölkopf, B. & Harmeling, S. (2010), Efficient filter flow for space-variant multiframe blind deconvolution, *in* 'Proceedings of the 23rd IEEE Conference on Computer Vision and Pattern Recognition'.

Joshi, N., Kang, S. B., Zitnick, C. L. & Szeliski, R. (2010), 'Image deblurring using inertial measurement sensors', *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010)* **29**(4), 30:1–30:9.

Kim, S.-J., Koh, K., Lustig, M., Boyd, S. & Gorinevsky, D. (2007), 'An interior-point method for large-scale ℓ_1-regularized least squares', *IEEE Journal of Selected Topics in Signal Processing* **1**(4), 606–617.

Klein, G. & Drummond, T. (2005), A single-frame visual gyroscope, *in* 'Proceedings of the 16th British Machine Vision Conference'.

Krishnan, D. & Fergus, R. (2009), Fast image deconvolution using hyper-Laplacian priors, *in* 'Advances in Neural Information Processing Systems'.

Krishnan, D., Tay, T. & Fergus, R. (2011), Blind deconvolution using a normalized sparsity measure, *in* 'Proceedings of the 24th IEEE Conference on Computer Vision and Pattern Recognition'.

Levin, A., Weiss, Y., Durand, F. & Freeman, W. T. (2009), Understanding and evaluating blind deconvolution algorithms, *in* 'Proceedings of the 22nd IEEE Conference on Computer Vision and Pattern Recognition'.

Mairal, J., Bach, F., Ponce, J. & Sapiro, G. (2010), 'Online learning for matrix factorization and sparse coding', *Journal of Machine Learning Research* **11**, 19–60.

Nagy, J. G. & O'Leary, D. P. (1998), 'Restoring images degraded by spatially variant blur', *SIAM Journal on Scientific Computing* **19**(4), 1063–1082.

Osher, S. & Rudin, L. I. (1990), 'Feature oriented image enhancement using shock filters', *SIAM Journal on Numerical Analysis* **27**(4), 919–940.

Sawchuk, A. A. (1974), 'Space-variant image restoration by coordinate transformations', *Journal of the Optical Society of America* **64**(2), 138–144.

Shan, Q., Jia, J. & Agarwala, A. (2008), 'High-quality motion deblurring from a single image', *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2008)* **27**(3).

Shan, Q., Xiong, W. & Jia, J. (2007), Rotational motion deblurring of a rigid object from a single image, *in* 'Proceedings of the 11th International Conference on Computer Vision'.

Shewchuk, J. R. (1994), An introduction to the conjugate gradient method without the agonizing pain, Technical report, Carnegie Mellon University.

Szeliski, R. (2004), Image alignment and stitching: A tutorial, Technical Report MSR-TR-2004-92, Microsoft Research.

Tai, Y.-W., Du, H., Brown, M. S. & Lin, S. (2010), 'Correction of spatially varying image and video motion blur using a hybrid camera', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **32**(6), 1012–1028.

Tai, Y.-W., Kong, N., Lin, S. & Shin, S. Y. (2010), Coded exposure imaging for projective motion deblurring, *in* 'Proceedings of the 23rd IEEE Conference on Computer Vision and Pattern Recognition'.

Tai, Y.-W., Tan, P. & Brown, M. S. (2011), 'Richardson-Lucy deblurring for scenes under a projective motion path', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **33**(8), 1603–1618.

Tibshirani, R. (1996), 'Regression shrinkage and selection via the lasso', *Journal of the Royal Statistical Society, Series B (Methodological)* **58**(1), 267–288.

Tomasi, C. & Manduchi, R. (1998), Bilateral filtering for gray and color images, *in* 'Proceedings of the 6th International Conference on Computer Vision'.

Vio, R., Nagy, J., Tenorio, L. & Wamsteker, W. (2005), 'Multiple image deblurring with spatially variant PSFs', *Astronomy & Astrophysics* **434**, 795–800.

Šorel, M. & Flusser, J. (2008), 'Space-variant restoration of images degraded by camera motion blur', *IEEE Transactions on Image Processing* **17**(2), 105–116.

Wang, Y., Yang, J., Yin, W. & Zhang, Y. (2008), 'A new alternating minimization algorithm for total variation image reconstruction', *SIAM Journal on Imaging Sciences* **1**(3), 248–272.

Whyte, O. (2012), Removing Camera Shake Blur and Unwanted Occluders from Photographs, PhD thesis, ENS Cachan.

Whyte, O., Sivic, J. & Zisserman, A. (2011), Deblurring shaken and partially saturated images, *in* 'Proceedings of the IEEE Workshop on Color and Photometry in Computer Vision (CPCV 2011), with ICCV 2011'.

Whyte, O., Sivic, J., Zisserman, A. & Ponce, J. (2010), Non-uniform deblurring for shaken images, *in* 'Proceedings of the 23rd IEEE Conference on Computer Vision and Pattern Recognition'.

Whyte, O., Sivic, J., Zisserman, A. & Ponce, J. (2012), 'Non-uniform deblurring for shaken images', *International Journal of Computer Vision* **98**(2), 168–186.

Xu, L. & Jia, J. (2010), Two-phase kernel estimation for robust motion deblurring, *in* 'Proceedings of the 11th European Conference on Computer Vision'.

Xu, L. & Jia, J. (2012), Depth-aware motion deblurring, *in* 'Proceedings of the IEEE International Conference on Computational Photography'.

Yan, J., Zhang, Y. & Yin, W. (2009), 'An efficient TVL1 algorithm for deblurring multichannel images corrupted by impulsive noise', *SIAM Journal on Scientific Computing* **31**(4), 2842–2865.

Zoran, D. & Weiss, Y. (2011), From learning models of natural image patches to whole image restoration, *in* 'Proceedings of the 13th International Conference on Computer Vision'.