Plenoptic Image Motion Deblurring
Paramanand Chandramouli, Meiguang Jin, Daniele Perrone and Paolo Favaro

Abstract—We propose a method to remove motion blur in a single light field captured with a moving plenoptic camera. Since motion is unknown, we resort to a blind deconvolution formulation, where one aims to identify both the blur point spread function and the latent sharp image. Even in absence of motion, light field images captured by a plenoptic camera are affected by a non-trivial combination of both aliasing and defocus, which depends on the 3D geometry of the scene. Therefore, motion deblurring algorithms designed for standard cameras are not directly applicable. Moreover, many state-of-the-art blind deconvolution algorithms are based on iterative schemes, where blurry images are synthesized through the imaging model. However, current imaging models for plenoptic images are impractical due to their high dimensionality. We observe that plenoptic images introduce periodic patterns that can be exploited to obtain highly parallelizable numerical schemes to synthesize images. These schemes allow extremely efficient GPU implementations that enable the use of iterative methods. We can then cast blind deconvolution of a blurry light field image as a regularized energy minimization to recover a sharp high-resolution scene texture and the camera motion. Furthermore, the proposed formulation can handle non-uniform motion blur due to camera shake as demonstrated on both synthetic and real light field data.

Index Terms—Plenoptic camera, light field image, motion blur, blind deconvolution.

I. INTRODUCTION

In the past few years, plenoptic cameras have entered into the realm of consumer photography [1], [2]. These cameras are equipped with 3D reconstruction and digital refocusing capabilities, not possible in traditional devices. This has led to an increased interest in the scientific community in high-quality reconstructions from light fields. As these commercial cameras are portable, camera shake is sometimes unavoidable and may result in blurry light field images. Similarly, moving objects can also cause the images to appear blurry.

Until now most of the research works on light field (LF) image processing have focused on depth estimation and super-resolution rather than motion deblurring. In contrast, motion blur in conventional cameras has been widely studied and current methods achieve remarkable results (see, for instance, [16], [48], [28], [9]). Unfortunately, the imaging mechanism of a conventional camera and LF camera are quite different. Due to the additional microlens array between the main lens and the sensors in the LF camera, an LF image consists of a rearranged set of views of the scene that are highly undersampled and blurred [30]. Consequently, motion deblurring methods that are applicable to conventional images cannot be adapted in a straightforward manner.

In this paper, we propose a motion deblurring scheme for images captured from a microlens-array based light field camera. From a single motion blurred LF image, we estimate the high-resolution sharp scene texture. An LF image can be related to the scene texture through a space-variant point spread function (PSF) which models the image formation mechanism. In addition, the sharp scene texture is related to the blurry texture in terms of a motion blur PSF. Modeling the LF image generation by taking into account these effects turns out to be very computationally intensive and memory inefficient. However, we show that, for constant depth scenes, it is possible to describe a motion blurred light field image as a linear combination of parallel convolutions (see sec. III-B). As a result, the model is extremely computationally and memory efficient. To model general 3D scenes, we consider them to be composed of layers having constant depth. We develop a fast GPU implementation that estimates high resolution scene texture from light field image for each depth layer. For each layer, we estimate the space-variant motion blur PSF through blind deconvolution. Finally, we solve for the sharp scene texture within an energy minimization scheme that accounts for alignment and distortion correction. We demonstrate the performance of our algorithm on real and synthetic images.

II. RELATED WORK

Since plenoptic image motion deblurring is related to image blind deconvolution as well as light field reconstruction, we provide a brief overview of research in these topics.

Single image motion deblurring Motion deblurring involves the joint estimation of a sharp image and a blur kernel. Because of its ill-posedness, motion deblurring algorithms enforce suitable priors for the sharp image and the blur kernel. With these priors, the motion blur kernel is initially estimated and finally the sharp image is recovered through non-blind deblurring [19], [27], [48], [41]. Popular choices of prior include Laplace distribution [28], total variation (TV) regularization [14] or $L_0$ regularization [50]. Other methods encourage sharp edges by using a shock filter [16], [48] or a dictionary of sharp edges [37]. For the blur function, the choices for prior include Gaussian distribution [48], or a sparsity-inducing distribution [19]. Levin et al. [27] have shown that a joint MAP estimation of the sharp image and blur function cannot lead to a correct estimate for a wide range of image priors. They show that marginalizing the sharp image and performing a MAP estimation on the blur function alone can correctly estimate the blur kernel. The marginalization of the sharp image is however computationally challenging, and therefore various approximations are used in practice [19], [28], [9]. Despite the theoretical analysis of Levin et al. [27], many methods successfully use a joint MAP estimation and
achieve state of the art results \[16\], \[33\]. Recently, Perrone and Favaro \[32\] have clarified this apparent inconsistency. They confirmed the results of Levin et al. \[27\] and showed that particular implementation details make many methods in the literature work and that they do not in practice solve a joint MAP estimation.

The aforementioned deblurring methods assume uniform blur across the image plane. Techniques exist that address non-uniform blur due to rotational camera motion by introducing additional dimensions in the motion blur PSF \[20\], \[23\], \[38\], \[45\]. Hirsch et al. propose an efficient implementation scheme wherein non-uniform blur can be modeled through piece-wise uniform blurs while still retaining the global camera motion constraint \[21\]. Nonetheless, it has been observed in \[26\] that in realistic scenarios, the performance of algorithms that explicitly handle rotational motion are not necessarily better than that of the methods that assume uniform blur. Motion blur can also vary within an image because of parallax in 3D scenes. Techniques proposed in \[35\], \[49\] address this problem when there are two observations available and the camera motion is restricted to in-plane translations. While in \[31\], a deblurring scheme which addresses non-uniform blur for bilayer scenes using two blurred images is proposed, Hu et al. address the problem with a single observation \[22\]. Other methods attempt to solve general space-varying motion deblurring by estimating locally uniform blur and carefully interpolate them to cope with regions having poor texture \[24\], or by iteratively employing uniform deblurring and segmentation algorithms \[25\].

**Light field capture** For a comprehensive discussion on light field acquisition, processing and display, one can refer to the recent survey in \[47\]. In this paper, we restrict our attention to microlens array-based plenoptic cameras. The basic lenslet-based plenoptic camera was first developed by Adelson and Wang for inferring scene depth from a single snapshot \[8\]. The portable design by Ng et al. with the use of microlens arrays triggered the development of handheld light field cameras \[30\]. The main drawback of image reconstruction in this camera was its limited spatial resolution. To overcome this problem, Lumsdaine and Georgiev designed the focused plenoptic camera wherein the microlenses were focused at the image plane of the main lens, thereby enabling rendering at higher resolution for certain depths \[29\]. Perwaß and Wietzke proposed a further modification in the design which included microlens arrays with microlenses having different focal lengths \[34\]. Recently, Wei et al. proposed to introduce irregularity in the lens design to achieve better sampling of light fields \[44\].

**Calibration, depth estimation and super-resolution** Since plenoptic images suffer from loss of spatial resolution, various algorithms have been proposed to super-resolve scene texture. In \[29\], the authors propose to render high resolution texture directly from light field data by projecting the LF image onto a grid at higher resolution than the number of microlenses. In \[11\], Bishop and Favaro model the image formation through geometric optics and propose a PSF-based model to relate the LF image and high resolution scene texture along with the camera parameters and scene depth. They propose a two-step procedure to estimate the scene depth map using view correspondences and hence the high resolution texture. Wanner and Goldlücke propose a technique to accurately estimate disparity using an epipolar image representation and then super-resolve 4D light fields in both spatial and angular directions \[43\]. Broxton et al. propose a 3D-deconvolution method to reconstruct a high resolution 3D volume using a PSF-based model with applications to light field microscopy \[12\]. We also found out that this work suggests a fast computational scheme that might be similar to the one proposed here. However, due to lack of details (a very short explanation is given in a paragraph in sec.3.4 of \[12\]) we cannot verify this. However, even in the presence of a similarity, our and this scheme were developed simultaneously \[5\]. Moreover, our scheme includes the case of motion blur.

Recently, many works have been published that deal with calibration, depth estimation and super-resolution \[17\], \[15\], \[39\], \[40\]. These methods are capable of dealing with practical issues in cameras such as noise, misalignments, vignetting etc. During the review of this paper, a light field deblurring algorithm was proposed by Srinivasan et al. \[36\]. They consider that the camera undergoes 3D translations and solve for camera motion and the sharp light field from a blurry light field image. They show that the estimation of scene geometry is not necessary for deblurring. The main difference between their paper and our work is that we aim to recover the high-resolution scene texture while they attempt to deblur the 4D light field function. The spatial resolution of the recovered light field in \[36\] corresponds to microlens resolution, which is very much coarser when compared to the texture that we recover in this paper. The difference in resolution is apparent in an example shown in Fig. 5.

**Contributions.** The contributions of our work can be summarized as: i) We introduce the problem of motion deblurring of light field images from a plenoptic camera and provide the first solution to it. ii) We propose a computationally and memory efficient imaging model for motion blurred LF images. iii) We will make publicly available a GPU implementation for recovering scene texture from light fields. iv) We solve a joint blind deconvolution and super resolution problem. v) We handle radial distortion correction and alignment within our energy minimization framework.

### III. Imaging model

In this section, we introduce notation and the image formation model for a motion blurred light field image. Initially, we describe the plenoptic image formation for constant depth scenario. We propose two approaches for fast implementation of the imaging model. Finally, we describe our model to represent non-uniform motion blurred scenes consisting of different depth layers.

#### A. Plenoptic image formation

Figs. 1 (a) and (b) explain the image formation process in a plenoptic camera. Consider a scene with depth value \(d\) and scene texture denoted by \(g_d\). Following \[11\], we consider \(g_d\) to be defined on the microlens array plane as shown in Fig. 1 (a).

Note that defining the texture on the microlens plane is only
Our website. In [12], [11] and efficient code to implement it is available on
of the plenoptic camera. An explicit formula is also available
The PSF


necessary to store
were of the order of mega pixels, the number of elements
For instance, if the scene texture and the light field image
challenging due to the memory and computation requirements.


B. Efficient LF image generation

A direct implementation of eq. (1) is practically very
challenging due to the memory and computation requirements.
For instance, if the scene texture and the light field image
were of the order of mega pixels, the number of elements
necessary to store \( H_d \) would be of the order of \( 10^{12} \).
Although \( H_d \) is sparse, performing the sum and product calculation in
eq (1) would still be computationally intensive. We address
these shortcomings by exploiting the structure of LF PSFs
due to the periodicity of the microlens array lattice. We show
two ways to arrive at exact and efficient implementations of
LF image generation that are highly parallelizable: first, via
parallel convolutions and second, through sparse matrix-vector
products.

We first provide an intuitive explanation of the key idea
that we exploit. Then, we present it formally. Fig. 2 (a) shows
an image captured by a Lytro Illum camera when a green
colored point light source was placed in front of the camera.
To better visualize the periodicity we synthetically overlaid
the hexagonal microlens array structure and set gaps between
microlenses to black. Small diagonal translations of the light
source yield images as in Figs. (b) and (c). Notice that the
image in Fig. 2 (c), is identical to that in Fig. 2 (a) except for
a translation of one microlens. Similarly, for every position of
the point light source, there exist corresponding positions (on
a discrete lattice) at which we observe repeated patterns.

To formalize this periodicity, we need to introduce some
basic notation. First, for simplicity, we consider a rectangular
microlens arrangement. The same formulation holds also for
hexagonal arrangements as shown in sec. III-C. Suppose there
are \( J \times J \) pixels under each microlens in the sensor plane.
A pixel location \( x \) can be written as \( x = kJ + j \), where
\( j = [j_1 \ j_2], \ j_1, j_2 \in [0, \ldots, J - 1], \) and \( k = [k_1 \ k_2], \)
where \( k_1 \) and \( k_2 \) are integers specifying the microlens under which
\( x \) falls. It should be noted that, while all those pixels with
the same \( j \) correspond to a view or a sub-aperture image (an
image from one camera in a camera array), those with the
same \( k \) correspond to the image within a single microlens.

We decompose the coordinates of texture \( p \) on the microlens
plane in a similar fashion. If two microlenses are \( D \) units
apart, then we write \( p = bD + t \), where \( t = [t_1 \ t_2], \ t_1, t_2 \in
[0, \ldots, D - 1], \) and \( b = [b_1 \ b_2], \) where \( b_1 \) and \( b_2 \) specify the
microlens coordinates. Note that \( D \) specifies the resolution
at which the scene texture is defined. A larger value of \( D \)
defines texture on a finer grid, and, vice versa, a smaller value
a coarser grid.

Now consider two points \( p \) and \( q \) located in different
microlenses in such a way that their relative position with
respect to the microlens center is the same (i.e., both have
the same value of \( t \)). Then, the PSFs \( h_d(x, p) \) and \( h_d(x, q) \) will
be shifted versions of each another. Technically, this is because
the PSF of the light field image is given by the intersections
of the blur discs of the main lens and the microlens array [11].
If there is a shift in the position of the point light source that
exactly corresponds to an integer times the microlens diameter,
then the resulting intersection pattern will be identical to the
original intersection pattern, but with a shift because of the
regularity of the microlens array as seen in Fig. 2. Hence,

\[
h_d(x, p) = h_d(x - Jy, p - Dy)
\]

for any \( y \), where \( y = [y_1 \ y_2], \ y_1 \) and \( y_2 \) are integers. Based

\[\text{http://www.cvg.unibe.ch/research/PlenopticDeblurring}\]
on this periodicity, we develop two approaches for efficient LF image generation.

**Convolution-based scheme:** The first observation is that the components of the PSF $h_d$ can be completely described by eq. (3) with $J^2 \times D^2$ kernels. Secondly, the spatial spread of every kernel will be limited because of the limited extent of the main lens blur disc. Thus, we have obtained a memory efficient representation of the PSF $h_d$. By replacing $p$ by $bD + t$ in eq. (1), and using eq. (3) we get

$$l_d(x) = \sum_t \sum_b h_d(x - bJ, t)g_d(bD + t)$$

(4)

Then, by expressing $x$ as $x = kJ + j$, we get

$$l_d(kJ + j) = \sum_t \sum_b h_d(kJ + j - bJ, t)g_d(bD + t)$$

(5)

Let $\hat{g}_d(b, t) = g_d(bD + t)$, $\hat{l}_d(k, j) = l_d(kJ + j)$ and $\hat{h}_d(k, t) = h_d(kJ + j, t)$ (i.e., just a rearrangement). Then, for every value of $j$, we have

$$\hat{l}_d(k, j) = \sum_t \sum_b \hat{h}_d((k - b)J + j, t) \hat{g}_d(b, t)$$

(6)

and finally

$$\hat{l}_d(k, j) = \sum_t \sum_b \hat{h}_d((k - b, t) \hat{g}_d(b, t)$$

$$= \sum_t \left( \hat{h}_d(\cdot, t) \ast \hat{g}_d(\cdot, t) \right)(k)$$

(7)

where $\ast$ denotes the convolution operation. Eq. (7) indicates that we can arrive at the LF image by performing convolutions for every possible value of $t$ and $j$. Note that these convolutions are completely independent and therefore can be executed in parallel. Also, we need only $J^2 \times D^2$ PSFs that are denoted by $\hat{h}_d(\cdot, t)$ for the LF image generation.

**Sparse matrix-based scheme:** The periodicity in eq. (3) can also be written as

$$h_d(x + Jy, p) = h_d(x, p - Dy).$$

(8)

By setting $x$ as $x = kJ + j$ and using eqs. (1) and (8), we get

$$l_d(kJ + j) = \sum_p h_d(j, p - kD)g_d(p).$$

(9)

Let $\hat{l}_d(k, j) = l_d(kJ + j)$ and $p' = p - kD$. Then we have,

$$\hat{l}_d(k, j) = \sum_{p'} h_d(j, p')g_d(p' + kD).$$

(10)

An efficient implementation of LF generation can be obtained from eq. (10). The term $\sum_{p'} h_d(j, p')$ denotes summing over a particular row of the basic PSF matrix $H_d$ (eq. (2)). Every row of the matrix $H_d$ is quite sparse since a pixel in a light field image is related to only a small number of elements of the scene texture. Consequently, a sparse matrix consisting of only $J \times J$ sparse rows, corresponding to all possible values of $j$, is sufficient to represent the PSF to render one micro lens. The image within a single micro lens (i.e., $l_d(k, j)$ for a particular $k$) can be obtained through the product of this sparse matrix with the texture elements $g_d$. For images corresponding to other micro lenses, one needs to apply only a translational shift $g_d(p' + kD)$ as seen in eq. (10). All these operations can be performed in parallel.

**C. Hexagonal arrangement**

In consumer plenoptic cameras, the micro lenses are arranged on a hexagonal grid. Fig. 3 (a) shows an ideal hexagonal arrangement of the micro lenses. The basis for our implementation schemes was that the intersection pattern of the main lens blur disc and the micro lens blur disc repeats periodically as we traverse the micro lens array plane. One can observe that the periodicity property holds when taking horizontal or vertical steps of $Q$ and $Q'$ pixels respectively, which match the dimensions of the red-shaded rectangular region in Fig. 3 (a). Similarly, as discussed in sec. III-B, in the case of micro lens arrays with a rectangular arrangement, the smallest periodic step corresponds to one micro lens with $J \times J$ pixels. We define the scene texture analogously. We consider that on the micro lens array plane there are $P \times P'$ units per block in the hexagonal arrangement case and $D \times D$ units in the rectangular case. This implicitly defines the view index $J = [j_1, j_2]$, $j_1 \in [0, \ldots, Q' - 1]$, and $j_2 \in [0, \ldots, Q - 1]$, and similarly, $t = [t_1, t_2]$, $t_1 \in [0, \ldots, P' - 1]$, and $t_2 \in [0, \ldots, P - 1]$. With these changes, based on our discussion in sec. III-B, one can see that the LF image generation model in eq. (1) can be implemented using either the convolution-based approach given in eq. (7) or the sparse matrix product-based scheme in eq. (10) even in the hexagonal arrangement case.

**D. Motion blur model**

A relative motion between the camera and the scene during the exposure interval causes the light field image to be motion blurred. In a general 3D scene, the averaging of a scene point at the camera image plane depends on the scene depth value. Consequently, image points corresponding to different depth values should be modeled separately. Let $\Omega_d$ denote the support of a layer in the texture domain with depth $d$, i.e., points $p \in \Omega_d$ have the same depth. For each depth layer, we assume the scene texture $g_d$ to be a blurry instance of the latent sharp image $f_d$. The subscript $d$ indicates that the values of $f_d(p)$ and $g_d(p)$ are valid only when $p \in \Omega_d$. Following the works in [18], [20], [23], [31], we consider blurring due to in-plane camera rotations and translations.
As derived in [18], the camera rotations and translations result in 2D transformations of the sharp image. Let \( g_d \) and \( f_d \) denote the vectorized forms of \( g_d \) and \( f_d \), respectively. The blur degradation can be represented in two equivalent ways: either as a sum of rotated and then filtered instances of the sharp texture, or as a linear combination of shifted and rotated textures

\[
g_d = \sum_{\theta \in \Theta} B_\theta R_\theta f_d = \sum_{\theta \in \Theta} F_{d\theta} \beta_\theta \tag{11}
\]

\( \theta \) denotes the angle of rotation within a set of angles \( \Theta \), \( R_\theta \) denotes the 2D warping matrix corresponding to the angle \( \theta \), \( F_{d\theta} \) denotes a matrix whose columns consist of different shifted instances of the rotated sharp scene texture \( R_\theta f_d \), and \( B_\theta \) is the matrix equivalent to performing a 2D convolution with the motion blur kernel \( \beta_\theta \). We also have the constraint that \( \sum_{\theta} \| \beta_\theta \|_1 = 1 \). The model described in eq. (11) is similar to the slice-based generalized additive convolution model described in [18].

We relate the light field image \( l_d \) and the sharp image \( f_d \) through a depth-dependent LF PSF and a motion PSF, separately for each layer support \( \Omega_d \). Note that this model does not capture occlusion effects at depth discontinuities. We consider that to be beyond the scope of our paper.

IV. PLENOPTIC MOTION DEBLURRING

In this section, we describe our methodology to recover a sharp scene texture from a blurry light field image. In our first step, for every depth layer, we recover the blurry scene texture \( g_d \) from the light field observation. We use this blurry texture to determine the motion blur PSF through non-uniform blind deconvolution. Finally, we determine the sharp scene texture for every layer through an energy minimization framework.

A. Texture recovery

Given a light field observation, we initially determine the scene depth map. In our first step, for every depth layer, we recover the blurry scene texture \( g_d \) from the light field observation. One could use any other depth estimation scheme as well. We apply the k-means clustering algorithm on the depth map to segment it into \( N_d \) clusters. The number of clusters is set by the user although we found 2–3 layers to suffice in most cases. As an example, for the raw image captured from a Lytro Illum camera shown in Fig. 4 (a), the depth map (top of Fig. 4 (b)) was segmented into two layers \( \Omega_1 \) and \( \Omega_2 \) as shown in Fig. 4 (b) (bottom).

For each depth layer, we evaluate the entries of the LF PSF using the depth value and camera parameters. We then determine the texture \( g_d \) from the LF observation by restricting the spatial support according to \( \Omega_d \). We minimize the following objective function through gradient descent

\[
\arg \min_{g_d} \frac{1}{2} \left\| H_d g_d - l_d \right\|^2_2 + \lambda \left\| g_d \right\|_{TV} \tag{12}
\]

where \( \left\| g_d \right\|_{TV} \approx \sum_p \left\| \nabla g_d(p) \right\|_2 \), with \( \nabla g_d = [g_{dx} \ g_{dy}]^T \), denotes the isotropic total variation of the scene texture [10], [13], and \( \lambda > 0 \) regulates the amount of total variation.

The gradient expression with respect to the data term \( E_{data} \) is given by

\[
E_{data} := \frac{1}{2} \left\| H_d g_d - l_d \right\|^2_2 , \quad \frac{\partial E_{data}}{\partial g_d} = H_d^T (H_d g_d - l_d) \tag{13}
\]

The transpose operation \( H_d^T \) can be efficiently implemented in an manner analogous to the forward operation described in eqs. (7) and (10). In our final implementation, we use the matrix-vector product version of eq. (10) because accelerated functions for GPUs that perform sparse matrix-vector product are easily available [4]. The gradient with respect to the TV prior is given by

\[
\frac{\partial \| g_d \|_{TV}}{\partial g_d} = \sum_{\theta} \beta_\theta \left( \frac{\sum_{p} \left\| \nabla g_d(p) \right\|_2}{\left\| g_d \right\|_2} \right) \left( \frac{\sum_{p} \left\| \nabla g_d(p) \right\|_2}{\left\| g_d \right\|_2} \right)^T \tag{15}
\]

We follow an approach similar to the projected alternating minimization (PAM) algorithm of [32]. At each iteration, we first update the current estimate of the sharp image \( f_d \) through a gradient descent iteration. The gradient with respect to the data term in eq. (14), which we denote again as \( E_{data} \), yields

\[
\frac{\partial E_{data}}{\partial f_d} = \sum_{\theta} R_\theta^T B_\theta \left( \sum_{\theta} B_\theta R_\theta f_d - g_d \right) \tag{15}
\]
Subsequently, we update the motion blur PSF represented by $\beta_0$ for all possible values of $\theta$. The gradient of the data term with respect to each $\beta_0$ is given by

$$\frac{\partial E_{data}}{\partial \beta_0} = F_{d\theta}^T \left( \sum_{\theta} F_{d\theta} \beta_0 - g_d \right).$$ \hspace{1cm} (16)

After the gradient update, we threshold negative values and apply a constant scaling for all values of $\beta_0$ so that $\beta_0 \geq 0$ and $\sum_{\theta} \|\beta_0\|_1 = 1$. For each depth layer $d$, the blind deconvolution is performed only at pixels corresponding to the layer support $\Omega_d$. To estimate an accurate motion PSF, we reduce the possibility of including pixels from other layers by shrinking the support of $\Omega_d$. This eliminates outliers due to depth discontinuities and errors in the estimated depth map.

Note that one could have also estimated motion blur directly from the light field image $I_d$ with the data term $\|H_d \sum_{\theta} B_{\theta} R_{\theta} f_d - l_d \|_2^2$. However, we observed that in such a scenario, the convergence would be much slower and would also involve significantly more computations.

C. Sharp texture recovery

Once the motion blur PSF for each layer is known, one could deblur $g_d$ to estimate the sharp scene texture. However, to improve reconstruction accuracy, we recover the scene texture directly from the light field observation rather than from the blurry texture $g_d$. According to our model, we can estimate $f_d$ for each layer by solving

$$\arg \min_{f_d} \frac{1}{2} \|H_d B f_d - l_d \|^2_2 + \lambda \|f_d\|_V.$$ \hspace{1cm} (17)

We define the motion blurring matrix $B$ as $B = \sum_{\theta} B_{\theta} R_{\theta}$ to simplify the notation. Finally, we modify our objective function to incorporate practical imaging aspects such as misalignments and radial distortion.

In real cameras the microlens array is not necessarily aligned with respect to the CCD sensor plane. This effect is apparent in Fig. 6 (a) which shows a crop of size 50x7728 pixels from a raw white image captured with the Lytro Illum camera. In Fig. 6 (a), the gap between adjacent rows of microlenses is tilted instead of being strictly horizontal. Furthermore, in real plenoptic images the distance between adjacent microlens centers in terms of pixels need not be an integer. Hence, we need to apply an affine mapping which would align the raw image to a regular grid model as shown in Fig. 3 with both $Q$ and $Q'$ as integers. We introduce a warping matrix $W$ that aligns the LF image $H_d g_d$ to the captured raw image. The affine transformation remains fixed for a camera and typically its parameters are stored as metadata [6].

At microlenses close to the corners of the plenoptic image we observe optical vignetting. This is because light rays entering the main lens at extreme angles hit surfaces inside the camera before reaching the sensor. Fig. 6 (b) shows an example of the vignetting effect on LF images. It contains a region cropped at the top left corner of a white plenoptic image. The image pixels under the influence of vignetting do not follow our image formation model. Hence, to avoid using the affected pixels, we use a mask $M$ generated by thresholding a white image. The entries of the mask $M$ will be zero at pixels that are affected by vignetting and one at other locations. Moreover, certain pixels in the sensor array...
always produce saturated intensities (hot pixels). We detect these pixels by imaging a dark scene and include them in the mask. We also mask the edge pixels of every microlens since they suffer from noise due to demosaicing [17].

Yet one more effect in real cameras is that of radial distortion. In the raw image shown in Fig. 4 (a), we can clearly observe barrel distortion (see, for example, the edges of the posters in the background). To account for this distortion, we define a transformation $C$, which radially distorts the motion blurred scene texture. For our experiments, we use the calibration technique available in [3] to determine the distortion parameters of our camera. By considering the effect of affine warping, masking and radial distortion, the modified objective function can be written as

$$\arg \min_{\mathbf{f}_d} \frac{1}{2} \| \mathbf{M} \odot (W H_d C B \mathbf{f}_d - \mathbf{L}_d) \|_2^2 + \lambda \| \mathbf{f}_d \|_{TV}$$

where $\odot$ denotes the Hadamard product (point-wise multiplication). The sharp scene texture for each layer is separately obtained via gradient descent. Finally, we merge these textures based on the support of each layer.

In our GPU implementation of the sharp texture recovery in sec. (IV-A), we did not include affine warping and radial distortion correction within the objective function. Affine correction was applied on the light field image to obtain an aligned light field image. This aligned image was used to recover the scene texture. Distortion correction was applied only once after recovering the scene texture. To highlight the importance of accounting for these changes, from the raw image of Fig. 4 (a), we estimated the scene texture according to the objective function in eq. (18) (without considering motion blur). In the resulting image shown in Fig. 5 (d), the text is more clearly readable than the Lytro rendered image in Fig. 5 (c) and our GPU-based result in Fig. 5 (b). Note that both Figs. 5 (b) and (d) were obtained at the same resolution while Fig. 5 (c) was rendered at a resolution close to our results.

D. Implementation details

We discuss the implementation details such as super-resolution factor, noise and computational aspects of non-uniform blind deconvolution.

Super-resolution factor: The number of texture elements per microlens (denoted by $D$ on rectangular grids and by $P$ and $P'$ on hexagonal grids) define the super-resolution factor. This factor defines both the resolution of the latent scene texture and that of the LF PSF $h_d$. In Figs. 5 (e), (f) and (g), we show the reconstructed texture for the resolution factors $P = 1$, $P = 2$, and $P = 3$, respectively. As the resolution factor increases, we see that the reconstructed texture tends to show more and more details. We also estimated texture for the factor $P = 5$ and $P' = 9$. However, we observed no improvement in the resolution quality with respect to the factor $P = 4$ and $P' = 7$ (Fig. 5 (d)). This is due to the inherent limit of the LF camera [11]. Therefore, in all our real experiments we reconstruct texture with resolution parameters $P = 4$ and $P' = 7$.

Noise: The raw images from commercially available LF cameras contain significant levels of noise [39], [40], [46]. Consequently, if the regularization is low, our reconstruction can lead to minor noise artifacts as seen in Fig. 5 (d). The effect of noise can be overcome by using a larger regularization value. In Fig. 5 (g), we set the regularization value to be $\lambda = 2 \cdot 10^{-4}$ and do not observe noise artifacts. In all our real experiments, we set $\lambda = 2 \cdot 10^{-4}$ for texture recovery.

Non-uniform blind deconvolution: For the successful convergence of blind deconvolution, we follow the commonly used strategy of multiscale implementation. Initially, both the 3D blur kernel and the image are represented in a coarse resolution. After a set of iterations of the PAM algorithm the image and blur kernel are upsampled to a finer resolution.

We select the set of angles $\Theta$ to be a uniformly sampled set centered around zero degrees. Following [45], the sample spacing is chosen such that it corresponds to a displacement close to one pixel at the farthest point from the center of rotation. For a typical choice of range $[-2^\circ, 2^\circ]$ and spacing of 0.12°, $\Theta$ would contain about 35 elements. In practice, only a small subset of these angles would be contributing to motion blur. Considering the whole set of $\Theta$ would be computationally quite intensive as well. Hence, we restrict the number of angles in $\Theta$ by defining a support [45]. At the coarsest level, the support includes the entire range of angles in $\Theta$. At the finer scales, the support is determined by ignoring angles with negligible weights. Similar to the strategy in [45], we update the support at each scale after upsampling the 3D blur kernel.

V. EXPERIMENTAL RESULTS

We evaluate our light field blind deconvolution algorithm on several synthetic and real experiments. For synthetic experiments, we use the images of the standard dataset [26] which contains non-uniform motion blurred images. From the motion blurred images, we generate the corresponding motion blurred light field data according to the PSF model. We perform real experiments with data captured from a handheld Lytro Illum camera. We use a fixed set of values for the camera parameters in synthetic as well as real experiments. These values are specified in the supplementary material.

A. Synthetic experiments

The dataset in [26] consists of realistic motion blurred images that were generated from a set of predefined camera trajectories. These camera trajectories were recorded from real
camera motions. We consider each image of the dataset as the blurry texture \( g_d \). For every image, we randomly select a depth value ranging from 60 cm to 3 meters, and generate an LF image. For the sake of comparison, we estimate the sharp texture with two different approaches. Both share the same initial processing: First, the motion blurred texture \( g_d \) is recovered from the LF image (see sec. IV-B), and then, through blind deconvolution of \( g_d \), we estimate the motion PSF. With this motion PSF, in one approach we estimate the sharp image \( f_d \) from the observation \( g_d \) by minimizing the data term \( \| \sum_\theta B_\theta R_\theta f_d - g_d \|_2^2 \) along with TV prior. This approach is referred to as two-step method. In the other approach, we use the same motion PSF to estimate the sharp texture directly from the light field observation \( l_d \) as in eq. (17), and we call it the unified approach. In both approaches the values of the TV regularization parameters were the same. The regularization weight was set as \( \lambda = 2 \cdot 10^{-6} \) for texture recovery, \( \lambda = 1.5 \cdot 10^{-3} \) for the alternating minimization scheme, and as \( \lambda = 2 \cdot 10^{-5} \) to perform the final deblurring.

In our LF PSF based model we cannot reconstruct the entire texture from an LF image due to boundary effects. There are twelve different blur kernels in the dataset of [26]. We consider only seven of these blur kernels (numbered from 1 to 7 in the

<table>
<thead>
<tr>
<th></th>
<th>PSNR ( \mu )</th>
<th>PSNR ( \sigma )</th>
<th>SSIM ( \mu )</th>
<th>SSIM ( \sigma )</th>
</tr>
</thead>
<tbody>
<tr>
<td>unified two-step</td>
<td>27.2</td>
<td>3.76</td>
<td>0.86</td>
<td>0.057</td>
</tr>
<tr>
<td>unified two-step</td>
<td>24.63</td>
<td>3.96</td>
<td>0.79</td>
<td>0.077</td>
</tr>
</tbody>
</table>

Fig. 7. Example images from the synthetic experiment. First column: motion blurred texture from the dataset of [26], second column: corresponding LF images, third column: two-step results along with estimated PSF, fourth column: results of the proposed unified deblurring, and fifth column: true texture.

Fig. 8. Depth and texture estimation from real motion blurred light field images: recovered texture (left) and depth map above the layer segmentation (right).
dataset) by excluding the ones with high spatial extent. We tested our algorithm on three different textures. Out of the 21 different scenes, three representative examples are shown in Fig. 7. The results on all the 21 scenes are shown in the supplementary material for visual evaluation. We evaluated the performance through Peak Signal-to-Noise Ratio (PSNR) and the Structural Similarity Index (SSIM) [42] metrics using the ground truth images provided by the authors of [26]. In Table I, we show the mean and standard deviation of the PSNR and SSIM metrics. The proposed unified method outperforms the two-step method. Note that, our PSNR values cannot be compared with that of conventional image deblurring since the input to our algorithm is a light field image in which the scene texture undergoes inherent aliasing and defocusing.

B. Real experiments

We perform real experiments using images of motion blurred scenes captured with a Lytro Illum camera. To read images from the Lytro Illum camera, we use the Light Field Toolbox V0.4 software [6] developed by the authors of [17]. We first normalize the intensity of the raw image by using the white image stored in the LF camera. We capture motion blurred observations of four different real scenes from a handheld camera. In addition to blurred observations, we also capture a separate sharp observation for a direct visual evaluation of our result. In all our experiments, the scene depth ranges between 60cm to a few meters and beyond. In all images we use the same camera settings. Even for obtaining the vignetting mask, we use the stored white image of the camera. After demosaicing the raw white image, the mask \( M \) was obtained by considering the pixel locations greater than the threshold value of 0.5 in all the three color channels.

**GPU based texture recovery.** The Lytro Illum camera has about \( 432 \times 540 \) microlenses and the raw image has about 15 pixels per microlens. We define a regular hexagonal grid with \( Q = 16 \) and \( Q' = 28 \) (sec. III-C) and apply an affine mapping on the raw LF image to align it to the microlens array structure. The scene texture is defined on a grid with \( P = 5 \) and \( P' = 9 \), which corresponds to about one-third of the full sensor resolution. We implement the texture recovery scheme discussed in sec. IV-A on an Nvidia K20 GPU. The execution time of one gradient descent iteration is about 0.25ms per layer for a single color channel. The algorithm easily converges in about 100 iterations even with an arbitrary initial estimate. When we implement the same task on an Intel(R) Xeon(R) core CPU, through the convolution-based scheme of eq. (7), the execution time is about 6.5 seconds per iteration of gradient descent. After texture estimation, we correct for the radial distortion and downsample to a texture grid with \( P = 4 \) and \( P' = 7 \). The downsampling operation
helps to reduce warping artifacts. The depth maps rendered by Lytro Desktop software, their segmentation, and texture estimation results for three out of four images of our dataset are shown in Fig. 8. The other image we use to test our algorithm is a motion blurred instance of the scene that we show in Fig. 4. For brevity, we avoid repeating a similar depth map and layer segmentation.

We estimate the motion PSF of each layer from the recovered textures (sec. IV-B). Subsequently, the sharp textures are estimated through non-blind deblurring (sec. IV-C). Finally, the layers are merged and color correction is applied by considering Lytro’s rendering as the reference. We compare the performance of the proposed algorithm with other conventional image blind deblurring techniques. These techniques are applied to the all-in-focus image generated by the Lytro Desktop software. This comparison is not entirely fair, because our scheme effectively makes use of the depth map while the conventional deblurring schemes are not able to use this information. We compare our method with the schemes in [16], [45], [50] since these techniques are known to work well even for non-uniform motion blur [26]. For the method in [50] we use the publicly available Matlab executable that also handles non-uniform motion blur [7].

The results are shown in Figs. 9-12. In all the examples, our method consistently renders a sharp estimate. The outputs from other techniques do achieve deblurring either in some specific regions or for a particular scene. However, on the whole, they fail to achieve deblurring due to factors such as depth changes and loss of information in the Lytro rendered image. In some cases, our results produce some artifacts due to inaccurate depth estimates. However, in most of the cases the results are artifact-free even at object boundaries.

VI. CONCLUSIONS

We have presented the first motion deblurring method for plenoptic cameras. Our method extends classical blind deconvolution methods to light field cameras by modeling the interaction of motion blur and plenoptic point spread functions. Moreover, our model is highly parallelizable and memory efficient. By following an alternating minimization procedure, we determine the unknown non-uniform motion blur. We estimate the sharp and super resolved texture by considering practical issues of alignment and radial distortion. Experiments show that our method successfully deals with non-uniform motion blur, and outperforms approaches based on conventional blind deconvolution.

A drawback is that our method is not applicable to scenes with continuous depth variations. Jointly handling motion blur, aliasing and defocusing effects of an LF camera, and depth
variations will be left to future work. Our method can be considered as a stepping stone towards this direction.

ACKNOWLEDGEMENTS

The authors thank the reviewers for their constructive suggestions. MJ and PF acknowledge support from the Swiss National Science Foundation on project 20002_1153324.

REFERENCES

Fig. 12. Deblurring results on the image of two persons with the PSFs shown in the insets.

---