Abstract—Most motion-blurred images captured in the real world have spatially-varying point-spread functions, and some are caused by different positions and depth values, which cannot be handled by most state-of-the-art deblurring methods based on deconvolution. To overcome this problem, we propose a depth-aware motion blur model that treats a blurred image as an integration of a sequence of clear images. To restore the clear latent image, we extend the Richardson-Lucy method to incorporate our blur model with a given depth image. The empty holes in the depth image, caused by occlusion or device limitations, are fixed by PatchMatch-based depth filling. We regard the depth image as a Markov random field and select candidate labels by using belief propagation to set and smooth depth values for empty areas. Deblurring and depth filling are performed iteratively to refine the results. Our method can also be applied to real-world images with the assistance of motion estimation. The deblurring process is shown to be convergent; moreover, the number of iterations and the level of noise amplification are acceptable. The experimental results show that our method can not only handle depth-variant motion blur but also refine depth images.

Index Terms—Deblur, depth-variant, Richardson-Lucy.

I. INTRODUCTION

Motion blur is a common problem in images captured by hand-held cameras. Blurring occurs when relative motion exists between the camera and the objects being recorded during exposure. Information in images may be lost due to motion blur. Thus, restoring clear latent images from a blurred input is critical. A simple assumption about motion blur is to regard a blurred image as the convolution result of a clear image and a point-spread function (PSF). The deconvolution problem can be solved by optimization or inverse Fourier transformation. However, the assumption of convolution with PSF simplifies motion blur into a spatially invariant problem, which is nearly impossible in real cases for two reasons. One is that motions such as zoom-in (zoom-out) or rotation will lead to different pixels having different motion paths. The other is that PSF varies with the depth values of objects, objects close to the camera suffer from more severe blurring than those far away. Some methods [1]–[4] attempt to segment an image into several regions and set a constant PSF for each region. However, these segments should be small enough to ensure that all regions have similar PSFs, which is neither suitable for complicated scenes nor conducive to pixel-wise deblurring. Although optimization methods can solve such problems by setting a different PSF for each pixel, the large number of pixels leads to time-consuming computation.

We propose a depth-aware motion blur model based on [5]. Our blur model treats a blurred image as an integration of a sequence of clear images. Each frame of the sequence is obtained by applying a transformation to the initial clear frame, which is from the transformation in 3D space. From the viewpoint of each pixel, it is implemented by multiplying the pixel’s coordinate with a transformation matrix. One assumption in our method is that the depth map obtained is at least less blurred compared to the color image, which could provide additional useful information in deblurring. With the assistance of the depth, the motion of each pixel is considered in three dimensions to simulate spatially variant PSFs, and the deblurring quality could be enhanced (see Fig. 1). As the transformation is applied to each pixel, spatially-varying PSFs

Fig. 1. Depth-aware motion deblurring. (a) Input blur image with spatially-varying motion blur. (b) Our depth-aware motion deblurring result.
are perfectly obtained. We extend the Richardson-Lucy (RL) method [6] and make it applicable to spatially variant motion blur in 3D space. One problem in considering the depth is that most depth images captured by current devices contain empty holes. In our deblurring model, each frame of the latent clear image sequence should have its corresponding depth map, which is obtained by applying the same transformation to the initial clear depth map. We use a PatchMatch-based method to fill these holes. Then, belief propagation is used to smooth the edges and reduce the noise in the intermediate results.

We separate the entire framework into two subproblems. In the depth filling, the latent clear image is set to be constant and is obtained from the intermediate results of certain deblurring iterations. In the deblurring, the depth image is given by the filling result. Deblurring and depth filling are performed iteratively to refine the clear images as well as the depth map. Fig. 2 shows the deblurring results of our approach for spatially variant blur with varying depth for the “dolls” scene. Our work makes the following three main contributions:

- We propose a motion blur model that takes depth into consideration and synthesizes blurred images with spatially variant blur caused by three-dimensional motion.
- We propose a new method to fill the empty holes in a depth image during blur synthesis, which can predict sharp and accurate edges in the depth image.
- We present a deblurring method for our blur model. Our deblurring can restore images suffering from pixel-level spatially variant blur. Moreover, it is robust against noise.

II. RELATED WORK

Depending on whether the PSF is given, image deblurring is categorized into two main types: blind deblurring (without PSF) and non-blind deblurring (with PSF). As the PSFs can be uniform or non-uniform, deblurring can be further categorized into spatially invariant or spatially variant problems. Most previous studies have focused on blur models with spatially invariant PSFs. Non-blind deblurring methods can restore latent images given the blurred images and the corresponding PSFs. In such cases, the blurred images can be synthesized with the ground-truth blur kernel. Traditional methods, such as the Wiener filter [7] and Richardson-Lucy algorithm [6] have attempted to minimize the difference between the convolution results of latent and blurred images. These methods suffer from ringing artefacts due to the loss of high-frequency details. The noise will also be amplified in the optimization and deconvolution. To overcome these problems, regulation terms are used to constrain the restored image. Chan and Wong [8] proposed the use of the L1 norm of the image gradient as the regularization term. Levin et al. [9] extended this method such that the regularization term can be any power of the image gradient as the Hyper-Laplacian term. In [10], threads were set and different terms were applied for different gradient values.

Compared to blind deblurring, non-blind deblurring involves the estimation of the blur kernel directly. Reference [8] employed the Laplacian regularization term for estimating the PSF. In [11], pit has been shown that the aforementioned method may yield a deblurring result that is the same as the source blurred image, and the PSF is a single point with a central value of 1. Thus, in [12], an approach for estimating the PSF according to the transparency of the blurred object edges was proposed. Recent methods, such as [10] and [13], combined deblurring and kernel estimation, and performed the two processes iteratively to obtain the final latent image. Fourier transformation [14] or Iteratively Re-weighted Least Squares (IRLS) [15] can be used to solve these optimization problems. However, these methods cannot handle images with spatially-variant PSFs.

The aforementioned methods use blurred image as the only image reference, and the deblurring quality varies. To obtain additional details about the latent image, a highly noisy clear image captured with a short exposure time was used to guide the deblurring [16]. In [4], two images with different resolutions were captured, and a low-resolution high-frequency image was used to guide the restoration of the high-resolution blurred images. In [17], three different camera models were used to improve the image quality. However, these methods require complicated devices. In [18], a method for sharp image estimation based on the depth-involved motion-blurred images is proposed. However, their method works on estimating the PSF for each pixel, they hence calculate a PSF shape for the whole image. Our approach focuses on restoring clear latent images by extending the Richardson-Lucy method to incorporate our blur model with a given depth image. Since depth maps may contain hole-regions caused by occlusion or device limitations, we propose hole-filling techniques via PatchMatch and loopy belief propagation with joint-bilateral filtering (JBF) for depth refinement to obtain smooth depth values for our depth-aware motion deblurring.

Fig. 2. Our deblurring result. (a) Input blur image with spatially-varying motion blur for objects with different depths. (b) Our result with depth refinement after every 10 iterations of intermediate image updating. (c) Ground-truth image. (d) Close-up view of the three images, with related depths on the right.
Recently, a method for estimating a spatially variant kernel was proposed [19]. In [1], image segmentation was used to set different PSFs for different layers. Tai et al. [20] used a hybrid camera framework and estimated one PSF per pixel, but this approach requires an auxiliary video camera. Machine learning was used in [21] and [22] to perform deblurring. However, the learning methods require the training and testing sets to have similar blur kernels; otherwise, the deblurring result is invalid. We propose a depth-aware motion blur model based on [5] in order to obtain pixel-level spatially variant PSFs. The depth values play an important role in improving the deblurring results, which makes it possible to simulate the motion path in three dimensions in the real world. Unlike [23], which uses depth to segment the image into layers, our method directly applies the depth value of each pixel to deblur that pixel. We also propose a new inpainting method for depth images based on PatchMatch and Markov random field (MRF) belief propagation to solve the problem of empty holes in real captured depth images. Although our method is a non-blind deblurring based on the RL algorithm [6], it can be extended to real cases with kernel estimation under user assistance.

### III. Approach Overview

Motion blur in real captured images is due to the relative motion between the objects being captured and the camera during the exposure time. The information for each pixel is the integration of the light incident on it during this period. The final image can be expressed as the integration of the light intensity with respect to time:

\[
I = \int_{0}^{T} I(t) dt
\]

where, \( I \) is the final image and \( I(t) \) is the light information at time \( t \). Note that \( I(t) \) may be infinite in real cases, which is not suitable for image processing. Hence, we represent it in discrete form as the sum of \( n \) image frames:

\[
I = \frac{1}{n} \sum_{i=1}^{n} I(i)
\]

where, all the light information in Eq. (1) is split into \( n \) parts. Each image frame is light integration result in a short period:

\[
I(i) = \int_{(i-1)T/n}^{iT/n} I(t) dt
\]

When the objects captured by the camera are static during exposure period, blurring occurs only because of the motion of the camera itself. With this assumption, the image information \( I(i) \) for each \( i \) can be regarded as the transformation result of the initial image information \( I(0) \). To be more general, \( I(0) \) is a three-dimensional image with depth information. Then, we can perform transformation from the 3D space to its projective space assuming perspective projection using a transformation matrix \( P \) [24], and further apply a homography \( H_i \) for the transformation of the projective space. Thus, we have our transformation matrix \( M_i = H_i P \), where \( H_i = \prod_{j=1}^{i} h_j \). Here, \( h_j \) is a homography defined by a \( 3 \times 3 \) nonsingular matrix up to a scalar, and we use a homography for its ability to model all planar transformations [5]. Therefore, \( M_i \) is the transformation using the 3D space and slice motion. Based on transformation matrix \( M_i \), each pixel in \( I(i) \) can get its corresponding position in initial image \( I(0) \):

\[
I(p, i) = I(M_i p, 0)
\]

In this equation, \( p \) is the coordinate of pixels in \( I(i) \) and \( M_i \) is the matrix that transforms \( I(0) \) into \( I(i) \) from the 3D space to the projective space. Through this representation, the blur image \( I_b \) can be obtained using only initial clear image \( I(0) \) and \( n \) transformation matrix \( M_1 \) to \( M_n \) as follows:

\[
I_b(p) = \frac{1}{n} \sum_{i=1}^{n} I(M_i p, 0)
\]

Fig. 3 compares our matrix-based blur model with the convolutional model. Our model generates the transformation matrices from the blur kernel. In this case, the deblurring problem becomes the problem of estimating the clear result \( I(0) \) given the blurred image \( I_b \) and all matrices \( M \). We will now show the advantage of our matrix-based blur model over the kernel-based model. The commonly used convolution form for image blur synthesis can be expressed as:

\[
I_b = I \otimes K
\]

where, \( K \) is the blur kernel, which is also called the point-spread function (PSF). This model assumes that all pixels pass through the same path during the exposure time, which is nearly impossible because of camera jitter. For example, it cannot synthesize camera rotation, in which case the center pixel remains static and the pixels in the four corners of the image move considerably. However, by using the matrix form...
Fig. 4. Approach framework for our deblurring method. Our approach takes the initial depth image and motion-blurred image as inputs. In each iteration, we use the current estimated latent image $I_t$ as the clear image to fill the empty areas in the initial depth image. Then, we compute a synthetic blurred image according to $I_t$, filled depth image, and transformation matrices. The error image is then estimated using the synthesized blurred image and the initial motion-blurred image. Using the estimated error, the latent image is updated to image $I_{t+1}$. Then $I_{t+1}$ is used to process the next iteration.

shown in Eq. (5), we can achieve translation, rotation, and scaling at the same time by using different matrices $M$.

Using matrix-based motion blur model, we propose our corresponding deblurring framework, as shown in Fig. 4. The input part includes depth image, blurred image, and blur kernel. Our model decomposes the blur kernel into several transformation matrices to synthesize the blurred image, as shown in kernel decomposition. To solve the problem of information loss in depth images, we use the depth image and blurred image as inputs to predict depth values of the empty holes in the initial depth map, as described in Section IV (See the MRF depth filling part in Fig. 4). In the depth refinement part, we use the intermediate deblurring result to refine the depth map, as described in Section IV-C. The deblurring part shows how we update the deblurring result iteratively. First, we synthesize the blurred image using the transformation matrix obtained from the input, as described in Section V-B. Then, we perform deblurring with the assistance of the depth image. The details of the deblurring model are given in Section V.

IV. MRF DEPTH FILLING

In general, we cannot obtain perfect depth image that provides depth values for each pixel. Some empty holes usually exist because of occlusion or limitations of depth cameras. Our method assumes the depth filling problem as an optimization problem that aims to minimize the following energy:

$$E_{total} = \sum_p E(p, D(p))$$  \hspace{1cm} (7)

where, $E(p, D(p))$ is the energy for pixel $p$ with depth value $D(p)$. For an 8-bit depth image, depth value $D(p)$ can have 255 choices. There are numerous pixels in an image, whose energies can affect one another. It is impossible to obtain an accurate solution for Eq. (7). Instead, to obtain the global minimum, we assume that each pixel is related only to its neighbor pixels and regard the image as a Markov random field (MRF). A MRF is an undirected graphical model that can encode spatial dependencies. Each pixel is represented as one node in this undirected graph and is connected to its four immediate neighbors, namely, up, down, right, and left (See Fig. 5). Under this assumption, the optimization problem for minimizing Eq. (7) can be solved by loopy belief propagation (LBP), which will be described later.

A. MRF Formulation

The energy of each pixel, $E(p, D(p))$ in Eq. (7) is defined to penalize the depth choice $D(p)$. It is affected by the difference between $D(p)$ and the refined depth value available in each iteration, which will be defined as data cost. In addition, the depth values in a small region are usually similar to one another; hence, we need to avoid a large depth difference between two neighbor pixels; this penalty is defined
as smooth cost. Thus, Eq. (7) can be defined as:

\[
E(p, D(p)) = dataCost(p, D(p)) + \lambda \cdot smoothCost(p, D(p))
\]  

(8)

In order to get the data cost and smooth cost for pixel \( p \) with label \( D(p) \) in Eq. (8), we need to define label \( D(p) \) first. To make the optimization problem sufficiently efficient, our approach allows each node to have at most eight labels (candidates). The candidates are selected by PatchMatch approach. For each node \( p \) with unknown depth value \( D(p) \), we let \( P(p) \) be its corresponding patch with radius \( r \) in the intermediate deblurred result. Then, based on the implementation of [25] according to the authors’ website (http://www.ee.iisc.ac.in/labs/cvl/research/imdenoising), the \( N \)-nearest patches for \( P(p) \) are obtained in the same color image. The depth label for patch \( P(p) \) can be obtain in two ways. The first way is to use the depth evaluation in the \( N \)-nearest patches as the label for \( P(p) \). Depth evaluation for a candidate patch is performed by the sum-and-average approach as follows:

\[
Depth(P'(p)) = \frac{\sum_{q \in P'(p)} D(q)}{\sum_{q \in P'(p)} isValid(q)}
\]  

(9)

where, \( P'(p) \) is a candidate patch from the \( N \)-nearest patches for \( P(p) \), \( D(q) \) is the depth value of pixel \( q \), and \( isValid(q) \) is set to 1 if \( D(q) \) is not equal to zero and set to 0 otherwise.

The disadvantage of the above method is that the evaluated depth of candidate patch \( P'(p) \) directly used as labels for the missing depth values of \( P(p) \) may cause inconsistency of depth intensities with neighbor depth values in depth map as shown in Fig. 6(b). This is because two patches with similar color information in the color image may be in two different depth layers of the depth map. We assume that the pixels in the same patch usually have similar depths, thus, we can use the relativity of depth information between two patches to adjust the evaluated depth by Eq. (9) as relative depth to be used as labels for missing depth of \( P(p) \) in the depth image, and generate a depth label with a candidate patch \( P'(p) \) as follows:

\[
Label(p, P'(p)) = \frac{\sum_{q \in P(p), q' \in P'(p)} D(q)}{\sum_{q \in P(p)} isValid(q)} \times Depth(P'(p))
\]  

(10)

where, \( q' \) is the corresponding pixel that maps \( q \) from patch \( P(p) \) to a candidate patch \( P'(p) \). The result using relative depth is shown in Fig. 6(c), and empty holes are now filled with more smooth depth values for depth filling effects. And through applying the PatchMatch method, we can obtain eight nearest patches and generate eight labels as required. For each label from a candidate patch \( P'(p) \), we use the data cost function to assign label \( Label(p, P'(p)) \) to node \( p \) as a penalty. The sum of absolute distances is used as the image distance in our approach. Thus, the data cost can be defined as:

\[
dataCost(p, D(p)) = \sum_{q \in P(p), q' \in P'(p)} ||I(q) - I(q')||
\]  

(11)

If we want to use color distance, then image \( I \) in Eq. (11) will be the intermediate deblur result. To use the depth distance, the refined depth image in each iteration is used instead. Experiments have shown that using the depth distance as the cost function can slightly improve the filling image in some regions; hence, it is used as the cost function in our approach.

In addition to the data cost, we use the smooth cost function to enforce local smoothing across adjacent nodes. Because we use MRF to solve the optimization problem, the smooth cost function for node \( p \) is only affected by its four immediate neighbor nodes, which we can defined as set \( \theta(p) \). Thus, the smooth function can be defined by the depth difference as:

\[
smoothCost(p, D(p)) = \sum_{q \in \theta(p)} ||Label(p, P'(p)) - D(q)||
\]  

(12)

Next, we will show how to use loopy belief propagation to minimize the energy function by data cost and smooth cost.

B. Loopy Belief Propagation

Using PatchMatch to get the candidates, each node can have at most eight labels. Although eight candidates can improve the performance considerably compared to searching through all 255 choices for depths, minimizing Eq. (7) to get the exact solution remains a time-consuming task. Loopy belief propagation (LBP) is an efficient dynamic programming approach, which is commonly applied as energy minimization solution for Markov random fields [26]. We utilize LBP [27] to rapidly obtain an approximate solution. LBP is performed...
Algorithm 1 Belief Propagation

1: BP(mrf, p, dir)
2: for valid depth labels i do
3:   for valid depth labels j do
4:     cost = dataCost(p, j) + smoothCost(i, j);
5:     for each direction sDir except dir do
6:       cost += msg(p_x, p_y).sDir, j);
7:     end for
8:     minCost = the minimum cost for all j;
9:     end for
10: newMsg(p_x, p_y, i) = minCost;
11: end for
12: update mrf with newMsg;

Algorithm 1 shows the details of this propagation function. Pixel p need to receive the message form the other three directions except dir. Pixels with different depth values will get different propagation messages. We use the minimum cost with neighbor pixels in the other three directions as the propagation message newMsg(p_x, p_y, d) for pixel p with depth value d. The cost is obtained by the data cost, smooth cost, and belief values in the previous iteration, as shown in Algorithm 1. Then, the belief value array is updated with the new message. For example, if we propagate toward the right, then mrf(p_x + 1, p_y).left, d) = newMsg(p_x, p_y, d). After propagation, the best candidate for each pixel is updated in function MAP(mrf), which is done using Eq. (13). After all iterations, the best choice of depth value will be used to fill the empty areas:

\[
best(p_x, p_y) = \arg\min_d \sum_i mrf(p_x, p_y).i, d)
\] (13)

We compared our initial depth filling results with the results of two other methods. The first method fills the empty areas using the nearest valid depth value on the same horizontal line. This method is fast and does not need any reference image. However, the filled result does not seem reasonable (See Fig. 8(b)). The other method performs depth filling by joint-bilateral filtering with the assistance of intermediate deblurred color images. The predicted depth value can be iteratively to converge to the final solution. In each iteration, a message is passed to adjacent nodes when all incoming messages have been received. In our method, there are four directions in total; hence, each pixel will receive a message from the other three directions before each propagation (See Fig. 7). We use a belief value array mrf to record message for each pixel, which is a four-dimensional matrix. Suppose we have a pixel p = (p_x, p_y), the term mrf(p_x, p_y).dir, d) denotes the belief value for pixel at position (p_x, p_y) with depth value d to its neighbor pixel in direction dir. Here, dir has four choices (up, down, right, and left), and the choices for d is equal to the maximum number of depth label candidates for each pixel. The belief propagation function BP(mrf, p, dir) is called when we propagate the message of pixel p to the direction dir.

Joint-bilateral filtering (JBF) can restore empty areas consistently. However, for large empty holes, this method cannot predict the edges well (See Fig. 8(c)). Our filling result is shown in Fig. 8(d). Fig. 8(f)–(h) shows the deblurring effects regarding hole fillings. It can be clearly seen from Fig. 8(f) that improper filling will cause deblurring artefacts, especially in hole regions. Fig. 8(g) shows that simply applying joint-bilateral filtering may address small hole, however, for relatively-large empty holes, this method cannot predict the edges well and may still cause problems in large hole regions for deblurring. Fig. 8(h) shows the deblurring method using our depth filling by LBP-based initialization, and our approach results in high-quality deblurring effects. Compared with Fig. 8(g), the improvement in deblurring for Fig. 8(h) is more pronounced in the whole neighboring region than just in the hole-region with depth filling by the LBP-based initialization. The reduction of the local errors in depth estimates will help improve the deblurring quality in each iteration, and the extent of the effects would improve on the deblurring in the whole participating region and globally (See the RMS error reduction for the ‘cones’ scene in Table I).
TABLE I
RMS Error Comparison for Deblurring Regarding Depth Filling

<table>
<thead>
<tr>
<th>Method</th>
<th>art</th>
<th>books</th>
<th>bowling</th>
<th>cones</th>
<th>dolls</th>
<th>moebius</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBP Depth Refinement</td>
<td>6.0058</td>
<td>5.3406</td>
<td>2.5775</td>
<td>6.7452</td>
<td>5.3989</td>
<td>3.9416</td>
</tr>
</tbody>
</table>

Fig. 9. Visualization of the 1D depth profiles. (a) The 1D depth profiles for the scanline in Fig. 6. (b) The 1D depth profiles for the scanline in Fig. 8.

We could see clearly from Table I that our LBP-based depth refinement is able to produce better deblurring results with the lowest RMS errors compared with other depth filling methods. Fig. 9(a) plots and visually compares the 1D depth profiles for the scanline in Fig. 6, and the 1D depth profile in Fig. 6(c) using relative depth values as labels shows best performance. Fig. 9(b) visually compares the 1D depth profiles for the scanline in Fig. 8, and the 1D depth profile in Fig. 8(d) shows best performance. The 1D depth profile in Fig. 8(c) is improper for the region of interest. Although the visualized 1D depth profile in Fig. 8(b) seems proper for the region of interest horizontally, yet it can be clearly seen from Fig. 8(b) that the depth values filled are obviously inconsistent vertically.

C. Depth Refinement

The LBP depth filling method described above is used to obtain the initial depth map with the blurred input as a reference. Because the intermediate deblurring results will improve as the number of deblurring iterations increases, our method uses intermediate results as references to refine the depth map. Our refining method is based on joint/cross bilateral filtering, the basic idea is shown in Eq. (14). The filtering is only implemented in areas that are empty in initial depth map. The two reference images used in joint/cross bilateral filtering are the intermediate result as the color reference and the LBP filling result as the depth reference. Our results indicate that the image quality of the depth map will improve with the quality enhancing of the color reference, as shown in Fig. 10.

V. DEBLURRING MODEL

In this section, we illustrate our proposed depth-aware motion deblurring used in the deblurring process. We first introduce the Richardson-Lucy (RL) deconvolution method [6] and then show how to apply it to our model. The remainder of this section will describe the implementation details of our approach, including blurred image synthesis, regularization settings, and an extension for real captured images.

A. Proposed Deblurring Approach

Image blur occurs when an ideal point source does not appear as a point but spreads out into what we called a point-spread function (PSF). The information for each pixel is the sum of many individual point sources. Using PSF and the latent image, we can express pixel $p$ in the blurred image using a convolution process as:

$$I_b(p) = c(p) = \sum_{q \in K} K_{pq} I_q$$

(16)

where, $I_b$ is the observed blurred image and $I$ is the latent clear image, $p$ and $q$ represent the pixel positions. $c(p)$ is a convolution process for a pixel located at $p$. $K$ is the point spread function and $K_{pq}$ represents the contribution of pixel $q$ in the latent image to pixel $p$ in the observed blurred image. Shepp and Vardi [28] have shown that we could use the likelihood probability for the Poisson distribution $P(I_b, K|I) = \prod_{q \in I} c(q) I_b(q) e^{-c(q)}$ to model the Richardson-Lucy method as a maximum likelihood solution, and $c(q)$ is a convolution process for a pixel located at $q$. As the matrix of the second derivatives of $P(I_b, K|I)$ is negative semidefinite, it is a concave function [28]. For the optimization of $P(I_b, K|I)$, we follow [29, Th. 2.19(e)], and the Karush-Kuhn-Tucker conditions are the sufficient conditions for $I$ to be a maximizer. For all $q \in I$, we deduce the iterative update rule for the Richardson-Lucy method as:

$$I(q) = \frac{\partial}{\partial I(q)} \ln \left( \prod_{q \in I} \frac{c(q) I_b(q) e^{-c(q)}}{I_b(q)!} \right) = 0$$

$$I(q) \sum_{q \in I} \frac{\partial}{\partial I(q)} (I_b(q) \ln(c(q)) - c(q) - \ln(I_b(q)!)) = 0$$

$$I(q) \sum_{q \in I} I_b(q) \frac{\partial}{\partial I(q)} c(q) - I(q) \sum_{q \in I} \frac{\partial}{\partial I(q)} c(q) = 0$$

$$I(q) \sum_{p \in K} \frac{I_b(p)}{c(p)} K_{pq} - I(q) \sum_{p \in K} K_{pq} = 0$$
Deblurring aims to calculate the most likely \( I \) given \( I_b \) and \( K \). Since \( \sum_{p \in K} K_{pq} = 1 \), by adding the iteration index \( t \) and using Eq. (16), we could then obtain Eq. (17) and iteratively determine the pixel value for \( I(q) \) as:

\[
I(q)^{t+1} = I(q)^{t} \sum_{p \in K} \frac{I_b(p)}{I_b(p)^{t}} K_{pq}
\]

where,

\[
I_b^t(p) = \sum_{q \in K} K_{pq} I^t(q)
\]

If we assume that the point-spread functions are the same for all the pixels, then Eq. (16) can be expressed in convolution form as shown in Eq. (6). Using convolution representation, we can transform Eq. (17) into Eq. (19) for the Richard-Lucy deconvolution as:

\[
I^{t+1} = I^t \times K \otimes \frac{I_b}{I_b^t \otimes K}
\]

To achieve pixel-level spatially variant blur, our motion blur model uses the matrix form for the blurred image, as shown in Eq. (5). Because the convolution form and the matrix form attempt to represent the same blurred image \( I_b \) given the clear image \( I \), we hence have:

\[
l^t(p) \otimes K = \frac{1}{n} \sum_{i=1}^{n} l^t(M_i p)
\]

where, \( l^t(p) \otimes K \) denotes the pixel value at position \( p \) for the convolution form. Now, we use another image, i.e., the error image \( l_e^t \) to show the error between the convolution result \( l^t \otimes K \) and the blurred image \( I_b \). We set \( l_e^t \) as:

\[
l_e^t = I_b \quad \frac{I_b}{I_b^t} = \frac{1}{\sum_{q \in \Omega} I^t(M_q)}
\]

Then, Eq. (19) can be applied for matrix form deblurring:

\[
I^{t+1} = I^t \times K \otimes I_e^t = I^t \times \frac{1}{n} \sum_{i=1}^{n} M_i^{-1} l_e^t
\]

Now we have transform convolution Richardson-Lucy method Eq. (19) into the matrix form Eq. (22). Hence, convolution is no longer needed. We also note that transformation matrix \( M_i \) is used only for matrix multiplication in the entire process. If we use a two-dimensional coordinate \((p_x, p_y, p_z)\) for the pixel position, then \( M \) will be a 3 x 3 matrix. Now, when we add the depth value \( D(p_x, p_y) \) to make the pixel position a three-dimensional vector \((p_x, p_y, p_z)\), where \( p_z = D(p_x, p_y) \), the deblurring process given by Eq. (22) is also valid once we use 4 x 4 transformation matrices \( M \) instead.

### B. Blurring Synthesis and Deblurring

Given the clear image and its corresponding depth image, we can set the coordinate for each pixel as \((p_x, p_y, p_z)\), where \( p_x \) and \( p_y \) are the pixel position in the clear image and \( p_z \) is the depth value in depth image. Then, we prepare several transformation matrices \( M \), of which 16 are sufficient to translate the initial image into several frames. By summing and averaging each frame, we can obtain the synthesis results, which suffer from the problem of information loss as shown in Fig. 11(a). We can see from Fig. 11(a) that there are some black holes in the image, which means that there is no corresponding information for these areas. This is because we only have the depth value for the pixels in each frame. We cannot determine all the scene details in the entire 3D space. We can fill the empty holes using interpolation:

\[
I(p) = \frac{\sum_{q \in \Omega(p)} w(q) I(q)}{\sum_{q \in \Omega(p)} w(q)}
\]

where, \( \Omega(p) \) is the set for valid pixels in the neighbor region of pixel \( p \), and \( w(q) \) is the weight for pixel \( q \), \( I(p) \) and \( I(q) \) are their intensity values. The filling result is shown in Fig. 11(b). With the synthesis result and the transformation matrices \( M \), we can now apply Eq. (22) to carry out the deblurring.

### C. Regularization and Depth Updating

Noise is inevitable in real captured photos. To smooth out the noise in the blurred images, we use the total variation (TV) regularization term introduced in [5], [8], and [30] in order to suppress the image noise when performing deblurring. We define \( P(I) \) as the prior probability of the recovered image \( I \). Therefore, we have the optimization function as:

\[
\text{arg max} \sum_{I} P(I) P(I_b, K) = \text{arg max} \sum_{I} P(I_b, K | I) P(I)
\]

\[
= \text{arg min}_{I} -\ln(P(I_b, K | I)) - \ln(P(I))
\]

\[
= \text{arg min}_{I} \sum_{q \in I} (c(q) - I_b(q) \ln(c(q)) + \lambda R(I(q)))
\]

where, \( R(I) = -\frac{1}{2} \ln(P(I)) \) is the regularization term used, and \( \lambda \) is the weight for the regularization. We calculate the first derivative for Eq. (24) according to \( I(q) \) and let it equal...
to 0, we hence have the following equation as:

$$\sum_{p \in K} \frac{I_b(p)}{\sum_{q \in K} K_{pq} I^t(q)} K_{pq} = 1 - \lambda \nabla R(I(q))$$  \hspace{1cm} (25)

where, \(\nabla R(I(q))\) is the first derivative of \(R(I(q))\). As \(t \to \infty\), we want to make the Richardson-Lucy method get convergence, \(I^{t+1}/I^t\) should equal to 1. Thus, using Eq. (25), the regularized Richardson-Lucy method could be expressed as:

$$I^{t+1}(q) = \frac{I^t(q)}{1 - \lambda \nabla R(I^t(q))} \sum_{p \in K} \left( \sum_{q \in K} I_b(p) K_{pq} I^t(q) \right) K_{pq}$$  \hspace{1cm} (26)

with its convolution representation as:

$$I^{t+1} = \frac{I^t}{1 - \lambda \nabla R(I^t)} \ast K = \frac{I^t}{1 - \lambda \nabla R(I^t)} \ast I_b = \frac{I^t}{I^t} \ast K$$  \hspace{1cm} (27)

Similarly, as shown in Section V-A, we could obtain the new regularized form using Eq. (26) as:

$$I^{t+1} = \frac{I^t}{1 - \lambda \nabla R(I^t)} \ast \frac{1}{n} \sum_{i=1}^{n} M^{-1} I_e^i$$  \hspace{1cm} (28)

The total variation (TV) regularization in [5], [8], and [30] is then applied for suppress noise during the deblurring. By replacing the regularization term \(\nabla R(I^t)\) in Eq. (28) with \(\nabla R_{TV}(I^t)\), we could obtain Eq. (29) as following:

$$I^{t+1} = \frac{I^t}{1 - \lambda \nabla R_{TV}(I^t)} \ast \frac{1}{n} \sum_{i=1}^{n} M^{-1} I_e^i$$  \hspace{1cm} (29)

where,

$$\nabla R_{TV}(I^t) = -\nabla \frac{I^t}{I^t}$$  \hspace{1cm} (30)

The solution introduced in [5] initializes \(\lambda\) as 1.0, and decreases it after every 100 iterations. However, in our experiments, we will restart the iterative procedure each time we update the depth image. Hence, we set \(\lambda\) to a small fixed value between 0.0001 to 0.001 (with image intensity varying from 0 to 1.0). In the depth filling process, the quality of the reference color image will affect the filled results. Hence, starting with the blurred image, we update the reference color image after a fixed number of iterations each time. In most cases, the iteration number will be set to 10. We do not update the reference image after each deblurring iteration because there is only a slight difference between two neighbor intermediate results. Frequent updating will be a time-consuming task.

D. Real Case Deblurring

In blurring synthesis, we decompose the blur kernel into several motion trajectories. Since the blur kernels in the real-life cases may have inconstant values, the decomposed trajectories might have different weights accordingly. However, in this paper, for problem simplicity, we utilize uniform motion assumption, which means the trajectories in this paper have the same weight. We have our transformation matrix \(M_i = H_i P\), where \(H_i = \prod_{j=1}^{i} h_j\). With the uniform motion assumption, all the trajectories in the paper will have the same weight, namely, \(h_1 = h_2 = \cdots = h_i\). Thus, \(h_j = \sqrt{T_i}, 1 \leq j \leq i\). Therefore, we can obtain \(h_j\) through calculating the \(i\)th matrix root of \(H_i\) [31], and estimation for the series of \(h_j\) is reduced to the estimation of \(H_i\). Although our approach is a non-blind deblurring method, it can be extended to images captured by 3D cameras. The additional task required for real case deblurring is motion estimation, which is essential for transformation matrix construction. In this study, motion estimation is performed with user assistance, which gives the positions of pixel pairs before and after motion. Suppose that the position set for pixels before motion is defined as \(B\) and that for pixels after motion is defined as \(A\). Obviously, \(A\) is the result of \(B\) multiplied by the transformation matrix \(M\). Hence, we can obtain the transformation matrix \(M\) by:

$$M = B^{-1} A$$  \hspace{1cm} (31)
The total motion will then be divided into \( n \) sub-motions represented by \( n \) transformation matrices. In our approach, we divided the motion into \( n \) parts. These \( n \) sub-matrices are then used to perform the deblurring as shown in Eq. (22). Because some empty holes remain in the depth map, the depth filling and depth refinement phases in our framework are still necessary. Fig. 12 shows the improvement achieved after our depth filling approach is applied to a real captured depth map.

**VI. EXPERIMENTAL RESULTS**

This section presents our experimental results, and the images we used are all obtained from the Middlebury stereo datasets (http://vision.middlebury.edu/stereo). RMS error analysis showed that our method would converge as the number of iterations increases. Comparisons between our method and other methods proved that our method could handle scenes with large depth differences. Moreover, we tested our method with noisy blurred images to demonstrate its robustness.

**A. Convergence Analysis**

Both the RL algorithm and the method introduced in [5] have been shown to be convergent. We tested our method in the same way to observe the changes in the RMS errors after each iteration. Because the number of iterations was large in the experiment, we updated the filled depth image every 10 iterations only over the first 100 iterations. Subsequently, this updating procedure would be repeated after every 100 iterations (instead of 10). The parameter \( \lambda \) was set to 0.0005 (color intensity varying from 0 to 1.0) for the regularization term in this example for all the iterations. Fig. 13(b)–(d) shows the deblurring results after different numbers of iterations. We defined the updating rate according to the similarity between two neighbor intermediate results. Fig. 14 shows the convergence of the RMS errors for four images with different levels of blur. We can see that the RMS decreases rapidly in the first 50 iterations. Most cases converge within 500 iterations according to our experiments. Moreover, the difference in the deblurring results is negligible after 100 iterations for images that are not blurred severely. We have performed the convergence analysis on an Intel® Core™ 4.00GHz CPU with 8GB memory over an image at the resolution of \( 1390 \times 1110 \) from the Middlebury datasets. We have performed our depth-aware deblurring for 1000 times to obtain the more meaningful average convergence time for stable purpose. Table II shows the running time of our convergence analysis. According to Table II, the whole convergence process costs about 36.04 seconds. Other operations including pre-processing and post-processing only take 0.23 seconds. Therefore, our method takes 36.27 seconds in total to obtain the final deblurring, which achieves high-quality deblurring at favorable speed.
**B. Comparison With State-of-the-Art Methods**

In this section, we will illustrate the difference between our approach and other deblurring methods. First, we compare our approach with the methods introduced in [5] and [13]. The former employs a deblurring model that is nearly the same as ours. Hence, transformation matrices are also used in their method, except the data for depth. We set the depth to 0.5 (with maximum depth value equal to 1.0) to simulate their results. Further, we used the total variant regularization term with $\lambda$ set to 0.0005 for both methods. The work of [13] is a blind deblurring method. Hence, we used our synthesized blurred image as the input, and the deblurring results were obtained using the software provided on their website with the default setting in CPU mode.

Fig. 15 shows the deblurring results of the three methods, with enlarged details for comparison. The first row is for the scene “bowling” scene, which suffers from large depth difference. While in the second row, i.e. the “cones” scene, the depth range is rather small. Finally, in the third row, i.e. the “art” scene, image blur is caused by rotation around the top-left corner. Compared to the blurring input in the first column, the results of [5] (second column) and the results of [13] (the third column) indicate better performance at the center of the scene. However, the background of the scene is over-deblurred for these two methods, resulting in some “white” pixels and areas in the background. In addition, the foreground regions lack deblurring, leading to obvious ringing artefacts. This is because the method introduced in [13] cannot handle spatially variant blurring. Hence, the entire image is deblurred with the same kernel estimated from the blurred image. Although the method introduced in [5] can handle spatially variant PSFs without considering depth information, it can only deal with this problem in two-dimensional image spaces. Thus, the inner and outer parts of the image will be deblurred in different ways. However, objects near to and far away from camera will not produce different results in the case of this method [5].

Our results are shown in the last column, and they indicate that both near and far areas in the scene are deblurred with higher visual quality, with fewer ringing artefacts in foreground and a clear, accurate background.

Fig. 16 shows the RMS errors of these methods in the three scenes: “cones”, “bowling” and “art”. The results of the method introduced in [13] indicate failure for the “bowling” and “art” scenes, with RMS errors greater than the input image. The results of the method introduced in [5] indicate failure for the “bowling” scene. The depth range in the “bowling” scene is large; hence, without depth information, these two methods [5], [13] cannot process the foreground and background differently and effectively. The blurriness in the “art” scene is caused by rotation, which is difficult for the method introduced in [13] to handle. However, our approach avoids such problems by considering the depth information and refining the depth maps during the deblurring processes.

We further compare our approach with three other kernel-based spatially invariant non-blind deblurring methods, namely RL deconvolution [6], the method of Shan et al. [10], and the method of Xu and Jia [32], in order to show the advantages of our motion blur model. Since our model is matrix-based, we need to generate a blur kernel from our transformation matrices. Fig. 17 shows the kernel estimated details. We select

---

**TABLE II**

<table>
<thead>
<tr>
<th>Depth</th>
<th>1st 10 iters</th>
<th>2nd 10 iters</th>
<th>3rd 10 iters</th>
<th>4th 10 iters</th>
<th>5th 10 iters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depth Map</td>
<td>5.89s</td>
<td>4.71s</td>
<td>3.23s</td>
<td>2.78s</td>
<td>2.16s</td>
</tr>
<tr>
<td>Deblurring</td>
<td>4.53s</td>
<td>2.83s</td>
<td>1.86s</td>
<td>1.39s</td>
<td>1.88s</td>
</tr>
<tr>
<td>Total</td>
<td>10.21s</td>
<td>7.59s</td>
<td>5.41s</td>
<td>4.10s</td>
<td>3.34s</td>
</tr>
</tbody>
</table>

---

Fig. 16. RMS errors comparison of our results with the results of Tai et al. [5] and Xu et al. [13]. The images used are “bowling”, “cones”, and “art”.

Fig. 17. Nine kernels estimated from blur images synthesized by our blur model (rotation around the top-left corner). We selected nine different positions in the image and generate point spread functions for these pixels.

Fig. 18. RMS error comparison between our approach and three other spatial-invariant deblurring methods: RL deconvolution [6], Shan et al. [10] and Xu and Jia [32]. The nine PSFs used for the three methods are shown in Fig. 17.
Fig. 19. Noisy images deblurring. The top row shows the noisy input blurred images. The second row shows our deblurred results. Gaussian noise is used with \( n \) set to 0 and \( \sigma \) set to 0.0002, 0.0004, 0.001, and 0.002 from left to right. RMS errors with the ground-truth clear images are given below each image.

nine positions in the image (left part of Fig. 17), and use the blur kernels of these nine pixels (right part of Fig. 17) as the kernel candidates. Then, we tested all nine kernels using the three spatially invariant deblurring methods. Fig. 18 shows the RMS errors of the three methods with the nine kernels given for the “art” scene under rotational motion (rotation around the top-left corner). Our approach is a spatially variant version that uses matrix transformation with a unique RMS error given on the right. The results show that our method produces a much lower RMS error than the three methods [6], [10], [32].

C. Noisy and Over-Blurred Images

We estimated the robustness of our approach by testing it on noisy images. We use the `imnoise` function in Matlab to add noise to our synthesized blurred images before deblurring. Gaussian noise was used in our experiment with a mean value of 0. The first row in Fig. 19 shows the noisy blur images with \( \sigma \) set to 0.0002, 0.0004, 0.001, and 0.002 respectively. Our deblurring results are shown in the second row. From the RMS errors, we can see that our method can handle low noise levels well and produces results with lower RMS errors than input noisy image. For noise with large \( \sigma \), our method does not amplify noise significantly, and the deblurring results remain reasonable. We also tested our approach with over-blurred image. However, for largely over-blurred cases as shown in Fig. 20(a), where the blur information is too complicated, our approach may not be able to fully recover a perfect deblurring and results in a failure case as shown in Fig. 20(b). But to some extent, the deblurred result Fig. 20(b) is still more visually pleasing than the original blurred image, which also means our depth-aware motion deblurring requires further development and more future work for further enhancement.

D. Real Examples

We use Intel® RealSense™ 3D Camera F200 to capture both the color and the depth image for real case deblurring. The RealSense™ Camera F200 is a stand-alone camera that can be attached to a computer, and can be applied for natural gesture interaction, face recognition, 3D scanning, etc. It contains mainly three key components, namely, a conventional camera, an infrared laser projector, and an infrared camera [33]. The infrared projector projects a grid onto the scene, and the infrared camera captures it to obtain depth maps. The RealSense™ Camera F200 has an effective range between 0.2m and 1.2m; the color camera supports resolution up to 1080p at 30FPS and 720p at 60FPS; the depth (infrared) camera supports depth resolution up to 480p at 60FPS and HVGA quality at 110FPS, and supports infrared resolution up to 480p at 300FPS [33]. Therefore, there is no need for capturing the depth maps separately using a tripod or other methods. The depth map capturing speed by RealSense™ Camera F200 is much faster than the capturing speed for the corresponding color image. In this way, utilizing a at least less blurred depth map as supplemental information for helping deblurring the more blurred color image is meaningful and would help promote the deblurring quality. Also, our work mainly focuses on depth-aware motion deblurring using loopy belief propagation, while the motion-blurred color images and the related clear depth maps are obtained using RealSense™ Camera F200. We used the UV maps provided in the SDK to map the depth map to the coordinates of the color image.

Real examples usually have complicated motions such as zoom-in (zoom-out) or rotation. Motion estimation for real captured images was performed with pixels pairs provided by users as shown in Fig. 21(a). Based on uniform motion assumption, easiest way to handle such uniform motion is
of pixel pairs. Each $h_j$ is calculated under uniform motion assumption. We compared our result (Fig. 21(d)) with [5] (Fig. 21(b)) and [13] (Fig. 21(c)). As can be seen from the results, without depth consideration, background was over-deblurred in Fig. 21(b). Our approach worked well for both the near and far areas. We note that our real example is not as perfect and significant as synthetic examples, which is mainly caused by estimation errors for $h_j$. Besides, motions in our real example are also not as huge as in synthesized blurred images. To evaluate the accuracy relation between user interaction and final deblurring results, we have performed quantitative evaluations on the synthetic examples with user-assisted markups of pixel pairs. Table III shows the Root Mean Square (RMS) error comparison for deblurring results with respect to the number of pixel pairs for user-assisted markups. Table III shows the Root Mean Square (RMS) error comparison for deblurring results with respect to the number of pixel pairs for user-assisted markups. We can see clearly from Table III that, by increasing the number of user-assisted pixel pairs, the RMS error reduces accordingly, and more than eight pairs of correspondence points could produce high-quality deblurring results.

<table>
<thead>
<tr>
<th>Markups</th>
<th>art</th>
<th>books</th>
<th>bowling</th>
<th>cones</th>
<th>doll</th>
<th>mobius</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 Pairs</td>
<td>6.3214</td>
<td>5.1131</td>
<td>2.4952</td>
<td>6.8317</td>
<td>5.6247</td>
<td>4.2136</td>
</tr>
</tbody>
</table>

**Table III**

RMS Error Comparison for Deblurring Results with Respect to the Number of Pixel Pairs for User-Assisted Markups

![Fig. 20](image1.png)

(a) Largely over-blurred image. (b) Failure case from our approach. Although the deblurring result is visually better than the over-blurred input, we still need to perform more further researches on our approach to provide better deblurring effects.

![Fig. 21](image2.png)

(a) Input blurred image with user-assisted markups of pixel pairs for motion estimation. (b) Result by [5]. (c) Result by [13]. (d) Our result.

**Fig. 21.** Comparison of results for real captured images. (a) Input blurred image with user-assisted markups of pixel pairs for motion estimation. (b) Result by [5]. (c) Result by [13]. (d) Our result.

**Fig. 20.** Failure case from too over-blurred image. (a) Largely over-blurred image. (b) Failure case from our approach. Although the deblurring result is visually better than the over-blurred input, we still need to perform more further researches on our approach to provide better deblurring effects.

presented by Shan et al. [34]. Their method relies on user to provide image correspondences to build the transformation in deblurring. Dai and Wu [35] also presented an effective method to estimate $H_t$ by blurred objects alpha matte. We utilize the method in [34] for real example experiments. Li et al. [36] further introduced sharp panoramas generation from motion-blurred videos, the method for estimating a global homography could also be applied as motion path estimation in this paper. In Fig. 21, transformation matrix $M_t$ is obtained by fitting the transformation matrix with user-assisted markups.

**VII. CONCLUSION AND FUTURE WORK**

It is difficult to restore a spatially variant blurred image, especially when each pixel has an individual point-spread function. Moreover, existing studies have not paid sufficient attention to the differences in blurriness caused by varying depth values. To address such issues, this paper proposes a matrix-based motion blur model to synthesize a blurred image with pixel-level spatially variant blur by using a depth image. We applied several transformation matrices using the 3D space and slice motion to produce intermediate frames and summed them up to form the blurred image. The problems of empty holes that occur in most depth images are fixed through our depth filling approach. We also propose an iterative approach to restore latent clear images using our blur model. The experimental results showed that our model can simulate image blur caused by three-dimensional motion, and the deblurring results are robust in different depth layers.

In the future, based on the model in this research, we will further work on image/video deblurring with nonuniform motion velocity assumption [37], [38]. We will further study and work on jointly optimizing the depth map [39]–[41] and clear image for the benefits of blur kernel estimation [37], [42] to propose a more general solution to handle blind image deconvolution. We also plan to improve depth filling quality for objects’ edges in order to make them sharper and more consistent with color reference images. Motion animation for real captured images is another important task, which would be fully automated and hence more effective than user assistance.

**REFERENCES**


Xiaoxin Fang received the B.Eng. and the M.Eng. degrees in computer science from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. His current research interests include image/video deblurring, real-time computer graphics, machine learning, and computer vision.

Ping Tan received the Ph.D. degree in computer science and engineering from the Hong Kong University of Science and Technology, Hong Kong, China. He is currently an Associate Professor with the School of Computing Science, Simon Fraser University, Burnaby, BC, Canada. His current research interests include computer vision and computer graphics. He has served as a Program Committee Member of SIGGRAPH and SIGGRAPH Asia.

Enhua Wu (M’12) received the B.Sc. degree from Tsinghua University, Beijing, China, in 1970, and the Ph.D. degree from The University of Manchester, Manchester, U.K., in 1984. He is currently a Research Professor with the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China. He is also an Emeritus Professor with the University of Macau, Macau, China. His current research interests include realistic image synthesis and virtual reality. He is a Member of the ACM and a Fellow of the China Computer Federation. He has served as a Chair or Co-Chair for various conferences, including SIGGRAPH Asia 2016. He is an Associate Editor-in-Chief of the Journal of Computer Science and Technology.