Temporal Coherence-based Deblurring Using Non-uniform Motion Optimization

Congbin Qiao, Rynson W.H. Lau, *Senior Member, IEEE,* Bin Sheng, *Member, IEEE,* Benxuan Zhang and Enhua Wu, *Member, IEEE*

***Abstract*—Non-uniform motion blur due to object movement or camera jitter is a common phenomenon in videos. However, state-of-the-art video deblurring methods used to deal with this problem can introduce artifacts, and may sometimes fail to handle motion blur due to the movements of the object or the camera. In this paper, we propose a non-uniform motion model to deblur video frames. The proposed method is based on superpixel matching in the video sequence to reconstruct sharp frames from blurry ones. To identify a suitable sharp superpixel to replace a blurry one, we enrich the search space with a non-uniform motion blur kernel, and use a generalized PatchMatch algorithm to handle rotation, scale, and blur differences in the matching step. Instead of using pixel-based or regular patch-based representation, we adopt a superpixel-based representation, and use color and motion to gather similar pixels. Our non-uniform motion blur kernels are estimated from the motion field of these superpixels, and our spatially varying motion model considers spatial and temporal coherence to find sharp superpixels. Experimental results showed that the proposed method can reconstruct sharp video frames from blurred frames caused by complex object and camera movements, and performs better than the state-of-the-art methods.**

***Index Terms*—video deblurring, non-uniform blur kernel, image deblurring, video processing.**

I. Introduction

VIDEO capture has become very popular in recent years, largely because of the widespread availability of digital cameras. However, motion blur is unavoidable in casual video capture due to camera shake and object motion during exposure. While camera shake may cause different degrees of frame blur, object motion can aggravate the spatiotemporal complexity of the blur. Thus, video deblurring is an important but challenging task in video processing.

As blur kernel estimation and image deconvolution are both ill-posed problems, most state-of-the-art deblurring methods may impose specific requirements, such as special hardware [1] to obtain motion, or assume that a blurry image is convoluted by a global blur kernel [2]. With multiple input images, the problem can be simplified by assuming a static scene [3] or one with few moving objects [4] [5]. While these deblurring methods may have good results under the specified conditions, they may not be able to represent the spatially varying motion due to camera shake and object motion using a single blur kernel, which is the key for deconvolution-based image/video deblurring methods. In addition, image deconvolution can cause additional artifacts due to small changes in noise or inaccuracy in blur kernel estimation. Thus, a method that can deblur casual videos is desired.

Since different frames as well as parts of a frame in a video sequence may have different degrees of sharpness, some sharp frames or regions of a frame may be used to restore other blurry neighbor frames or regions based on temporal coherence. In [6], blurred-unblurred frame pairs in a video are used to refine blur kernel estimation. However, this method still suffers from the same limitation as deconvolution-based methods. In [7], sharp frames are used to replace blurry frames based on path-based synthesis, and blurry artifacts in a scene without moving objects or with only minor object movement are removed. However, such homography-based methods may fail to handle large-scale object movements, which typically involve complex scale/rotation transformations. Although some generalized homography-based methods [8] [9] may be able to handle large-scale motions better, they are still restricted to the image as a whole and cannot handle object motions in video frames.

In general, objects in a video likely have different motions. A global motion model or a limited, spatially varying motion model, such as the homography-based motion model, cannot represent actual motion blur. In this paper, we propose a non-uniform motion blur model to solve this problem. First, we use an adapted Simple Linear Iterative Clustering (SLIC) segmentation algorithm to divide each image into superpixels by considering image texture and motion, which preserves object outlines better than regular patch-based image representation. Second, we compute a non-uniform motion blur model, estimated by the motion vector of each superpixel, to approximate object motion in each blurry frame. Third, we search for a sharp version of each blurry superpixel temporally (i.e., from a sharp neighbor frame) and spatially (i.e., from the same frame) using a local blur kernel. We then apply a texture synthesis algorithm to reconstruct the deblurred frame to avoid introducing blurry or blocky artifacts.

This paper makes two main contributions. First, we propose a non-uniform motion model to represent the movements of superpixels based on the estimated motion field. Second, we improve the PatchMatch algorithm [10] [11] using the motion blur kernels, and extend the search both spatially and temporally to find the most appropriate sharp superpixels. Experimental results show that our non-uniform motion model produces more accurate deblurred frames than state-of-the-art methods and avoids generating extra artifacts, especially ringing artifacts.

II. Related Work

Image or video deblurring has been extensively studied, and many proposed methods have yielded great success. Although a blurry image can be sharpened by convolution through different point-spread functions (PSF) or blur kernels, e.g., spatially varying PSFs, restoring a blurry image is inherently an ill-posed problem. While video deblurring can make use of extra information from adjacent frames in the deblurring process, it must also consider additional problems, including spatial and temporal coherence. Here, we summarize the main studies on image and video deblurring.

Single-image deblurring. Single-image deblurring: This problem has been extensively studied, and a number of approaches have recently been proposed. Most of them are based on a uniform blur model with different image priors [12] [13]. The imposed assumptions on the motion blur, e.g., the entire image having a uniform blur, make these methods difficult to be applied directly to video frames, which typically involve complex motion blur.

There are methods that attempt to estimate spatially varying motion blur. Whyte *et al.* [14] estimate a non-uniform blur kernel for 3D camera rotation, but it does not consider motion blur caused by object movements. Xu *et al.* [15] proposed a motion deblurring model based on unnatural representation, which can be applied to uniform and non-uniform motion blur. However, since it only considers camera rotation and translation, it shares the same limitation as [14]. Levin [16] proposed a deblurring approach for images with few moving objects. It assumes that only part of the image is blurred such that the image can be segmented into blurred and unblurred layers, and uses image statistics to distinguish and correct the motion blur against a stable background. Shan *et al.* [17] proposed a rotational motion blur model using a Fourier transform with an estimated transparency map. The motion kernel is space variant but needs user interaction to indicate object motions. Dai and Wu [18] proposed an α-motion blur constraint model to estimate a blur kernel for each pixel based on matting extraction. However, it relies strongly on the accuracy of the mattes. Kim and Lee [19] proposed an energy model to estimate the motion flow and the deblurred image based on the robust total variation (TV)-L1 model. This method can handle abrupt motion changes without segmentation.

As these single-image deblurring methods focus on deblurring a single image, directly applying them to recover blurry videos frame by frame cannot guarantee temporal consistency.

Multi-image deblurring. With multiple input images, more information can be obtained to help improve the quality of the deblurred results. Rav-Acha and Peleg [20] use two blurry images in different blur directions to obtain deblurred results superior to those of the single-image deblurring methods. Some methods are proposed recently to reconstruct a shape image from two or more blurry images by deconvolution using different motion kernels, e.g., [2] [21] [22]. However, all these methods assume a global motion kernel for each image and share the same limitation as the uniform motion model in single-image deblurring methods. To handle non-uniform motion blur introduced by moving objects, Onogi and Saito [4] proposed a video sequence restoration method by segmenting a blurry image into background and foreground based on moving objects. However, it assumes that the blur kernels estimated by the motion are accurate, and uses them to directly deblur the image. Cho *et al.* [5] segment a blurry image into background and a few moving objects with similar motion. Their spatially varying motion blur kernels were defined by a uniform motion blur of the moving objects. However, real object motion is typically non-uniform. Tai *et al.* [1] proposed a non-uniform motion model using a hybrid camera system. They estimate spatially varying motion blur kernels using optical flows. While our method is similar to [1], it does not require a special camera. We compute optical flows from adjacent frames in order to estimate non-uniform motion blur kernels. Although the estimate kernels are not as accurate as those of [1], they are sufficiently robust to model motion in the blurry frame with our deconvolution-free deblurring.

Video deblurring. This typically involves more complex motion. Matsushita *et al.* [23] proposed an interpolation-based video deblurring method. They align a sharp frame and a blurry frame and replace the blurry pixels using the sharp ones. However, they do not consider the motion difference between these frames. Li *et al.* [3] proposed a deblurring method using a homograph-based model in their panorama generating system. Since they consider the homograph-based motion model as an accurate blur kernel, they fail to deconvolute moving objects. He *et al.* [24] proposed to segment moving objects into small regions and track their corner points across frames. By interpolating the motion within each region, spatially varying blur kernels can then be computed for deblurring. While this method proposed a new way to address the accuracy problem of optical flow on blurred videos, relying on the detection of shape corner points across frames is not reliable, e.g., under camera motions. Takeda and Milanfar [25] proposed a shift-invariant space-time point spread function (PSFs) for video deblurring. Its advantage is that no object segmentation or motion information is needed. However, it requires the camera parameters to be fixed throughout the video and the spatial and temporal PSFs to be known, which may not be practical for most applications. Lee *et al.* [6] proposed a residual deconvolution deblurring method using a blurred-unblurred frame pair. They iteratively refine the blur kernel to reduce ringing artifacts caused by conventional deconvolution. However, their estimated blur kernel is uniform for a frame and cannot handle complex motion blur.

In [26], Wulff and Black proposed a layered model to jointly estimate parametric motion and scene segmentation. This layer model separates a frame into a few layers. In each layer, it can only handle one or two moving objects. However, this affine model cannot approximate motion blur in complex scenes and has difficulties in handleing boundaries between layers. Delbracio and Sapiro [27] proposed a uniform deblurring method based on first consistently registering adjacent frames and then fusing them blockwise in the Fourier domain. While this method is very efficient, it assumes the input video to have a uniform motion blur and therefore aims at handling only camera shakes. Kim and Lee [28] proposed a single energy

function to estimate the optical flows and deblur the frames. While this method can handle non-uniform motion blur, it requires minimizing a complex energy function. On the other hand, our proposed method tries to restore the image through replacing blurred superpixels with sharp ones obtained from other frames. Cho *et al.* [7] proposed a deblurring method based on the motion model proposed in [3], which uses patches from nearby sharp frames to restore blurry frames through texture synthesis. Although this method can effectively reconstruct blurry videos, it has two limitations. First, it assumes homography-based motion, which cannot handle diverse object movements, as mentioned earlier. Second, their regular patch-based search and synthesis do not distinguish between moving objects and background. On the contrary, we propose a non-uniform motion model to handle motion blur due to camera and object movements. Although our method also involves patch-based deblurring, we consider motion in segmentation.

In summary, our method differs from previous work in that our main goal is not only to handle blurs caused by a shaky camera, but also complex blurs due to moving objects.

III. Frame Deblurring Algorithm

Our approach is based on the assumptions that the blur kernel varies spatially in a frame and temporally among frames, and sharp frames or superpixels are dispersed in the entire video sequence. We let $I_0, \ldots, I_n$ be the input video frames, and $L_0, \ldots, L_n$ the latent frames to be reconstructed. We also let I_i be a blurry frame, and I_j an adjacent frame with sharp superpixels as candidates to replace the blurry ones in I_i. Our goal here is to reconstruct L_i, a sharp version of the blurry frame I_i in the video sequence. Thus, our method needs to find proper sharp superpixels in I_j to replace the blurry superpixels in I_i. We propose a PatchMatch-based search strategy with spatial and temporal coherence to find these sharp superpixels.

Fig. 1 shows an overview of the proposed deblurring method, which has four steps. First, we compute the motion field to describe pixel motion between adjacent frames (Section III-A). Second, we propose a motion-based segmentation method to segment each frame into superpixels (Section III-B). Third, we present our non-uniform motion blur model to describe the motion of each superpixel in each frame (Section III-C). Finally, we present our PatchMatch-based search method, which uses the non-uniform motion model to find sharp superpixels, and our superpixel synthesis strategy, which replaces blurry superpixels and smoothes superpixel boundaries (Section III-D). Algorithm 1 summarizes our method.

Algorithm 1 : Summary of the proposed method.

```
for each frame I_i do
    compute the motion field;  (Sec. III-A)
    use revised-SLIC to get superpixel set φ_i;  (Sec. III-B)
    for each superpixel I_i^t ∈ φ_i do
        estimate blur kernel for I_i^t;  (Sec. III-C)
        use generalized patchmatch to find sharp superpixel;
        (Sec. III-D: Eq. 9 to 12)
        Replace blurred superpixel with sharp superpixel
        through texture synthesis;  (Sec. III-D: Eq. 13)
    end for
end for
```

A. Computing the Motion Field

As the motion field of two adjacent frames roughly indicates the trajectories of local motions during the exposure time, we compute the motion field based on optical flow [29]. For each pixel in frame I_i, we calculate the backward motion vector from I_i to I_{i-1} as (u^0, v^0), and the forward motion vector from I_i to I_{i+1} as (u^1, v^1), where u and v are the horizontal and vertical motions. The motion vector for a superpixel is then the average of all pixel motion vectors within the superpixel. This motion field plays a number of important roles in our motion model. First, our segmentation method uses it to gather pixels of similar motion to form superpixels. Second, we use it to estimate spatially varying motion blur. Third, we use it to compensate for the misalignment among neighbor frames to locate the search window.

The degree of motion blur of a pixel can be approximated by the displacement distance during the exposure time. As in [7], we estimate the sharpness of pixel p using the forward and backward motion vectors, which approximate the displacement of p between adjacent frames:

$$\alpha_p = \exp\Big(-\frac{(u_p^0)^2 + (v_p^0)^2 + (u_p^1)^2 + (v_p^1)^2}{4\delta^2}\Big), \quad (1)$$

where δ is a constant such that α_p ranges from 0 to 1. δ is set to 10 in our implementation. The sharpness of a frame or a superpixel is defined using the average sharpness value of all pixels within it.

B. Motion-based Segmentation

Simply dividing a frame into regular patches may increases matching errors when finding sharp patches from other frames. Here, we adopt the SLIC algorithm [30] to segment each frame into superpixels. The original SLIC algorithm is an adaption of k-means clustering for superpixel generation with a small search space, and takes color and spatial distance into account in the segmentation to preserve object boundaries. Our motion-based segmentation method adds the motion vectors to distance metric for the following two reasons. First, the motion vectors can describe local motions between adjacent frames and thus help distinguish different moving objects. Second, the motion vectors are used to estimate the motion blur kernels, and therefore the amount of deblurring. Hence, the distance between two pixels p and q is computed as:

$$D(p,q) = \sqrt{\Big(\frac{d_c(p,q)}{N_c}\Big)^2 + \Big(\frac{d_s(p,q)}{N_s}\Big)^2 + \delta\Big(\frac{d_m(p,q)}{N_m}\Big)^2}, \quad (2)$$

where $d_c(p,q)$ and $d_s(p,q)$ are the color and spatial distances, respectively, between p and q. N_c and N_s are the maximum color and spatial distances, respectively. We define N_c seperately for each superpixel from the previous iteration to maintain a superpixel size of approximately $N_s = 25$ (the

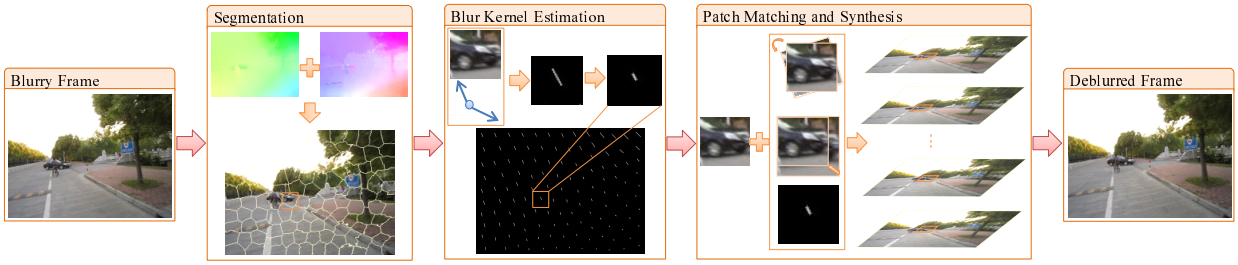

Fig. 1. Overview of the proposed video deblurring method.

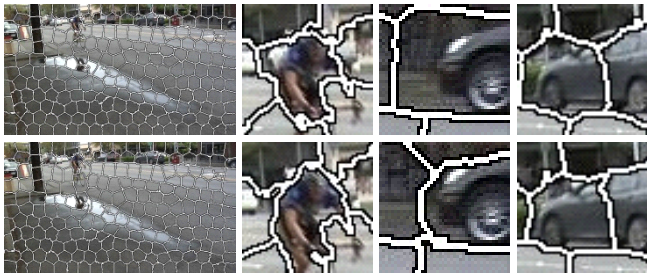

Fig. 2. Comparing SLIC superpixel segmentation with our motion-based SLIC segmentation. The first row shows the zero parameter version of SLIC [30]. The second row is our motion-based SLIC. Our segmentation can distinguish moving objects and segment blurry images better.

average superpixel size in our implementation). $d_m(p,q)$ is the motion distance between p and q and is defined as:

$$d_m(p,q) = \sqrt{(u_p^0 - u_q^0)^2 + (v_p^0 - v_q^0)^2 + (u_p^1 - v_q^1)^2 + (v_p^1 - v_q^1)^2}, \tag{3}$$

where (u_p^0, v_p^0) and (u_p^1, v_p^1) are the motion vectors for p, while (u_q^0, v_q^0) and (u_q^1, v_q^1) are those for q. As with the zero parameter version of SLIC, we set the maximum motion distance N_m adaptively for each superpixel using the maximum d_m value obtained from the previous iteration.

To set the effect of the motion field on segmentation, we introduce δ in Eq. 2. A larger δ can distinguish pixels with different motions, but may also destroy object structure. Our experiments show that setting $\delta = 3$ provides a good balance among color distance, spatial distance and motion in the segmentation. Fig. 2 compares our motion-based SLIC with the zero parameter version of SLIC. We see that motion information helps generate more accurate segmentation in moving objects. After segmentation, I_i can be represented as a superpixel set Φ_i , and we use $I_i^t \in \Phi_i$, to denote a superpixel.

Note that a blurry frame may contain mostly blurry superpixels. The blur becomes much severe if a superpixel represents a fast moving object. Even though we have considered the motion vectors, it may still be difficult to accurately segment a significantly blurred frame into superpixels. However, unlike other region-based image deblurring methods [4] [5], which rely on accurate segmentation of moving objects, we only need to obtain superpixels that can roughly separate different motion parts and then estimate a blur kernel for each superpixel for matching. Since our superpixels are small (around 25×25 pixels), only a small amount of edge information is enough to improve the accuracy of superpixel matching.

C. Non-uniform Motion Blur Kernel Estimation

After a frame is segmented into superpixels, we estimate a blur kernel for each superpixel and combine the blur kernels of all superpixels to form the spatially varying motion blur kernels. To minimize estimation errors due to severely blurred moving objects, we optimize the blur kernels from neighbor superpixels to find globally smoothed blur kernels. The estimated motion blur kernels serve as the initial blur filter in the superpixel matching step.

We compute the average backward and forward motion vectors for each superpixel $I_i^t \in \Phi_i$, as (u_t^0, v_t^0) and (u_t^1, v_t^1), respectively. In a video, the motion of a pixel across frames can be described by a complex motion kernel, which is difficult to estimate. A motion vector indicates total displacement during the interval between one frame and the next. In general, the video frame rate is over $24fps$ and the time interval should be smaller than $40ms$. As the exposure time should be shorter than this interval, we can simplify the blur model by assuming constant velocity motion. Like [3] [17], we integrate the per-superpixel motion vectors to form spatially varying blur kernels. Since the motion vectors obtained in Section III-A are between two frames, we need to estimate the motion vector during the exposure time, which is often less than $20ms$. We thus compute a ratio between 0 to 0.5 for an optimal value.

To estimate the motion blur for all superpixels in ϕ_i, we needs to minimize the following error:

$$E_{blur} = \min_{\theta} \sum_{I_i^t \in \Phi_i} (\|L_i^t \otimes K_i^t(\theta) - I_i^t\|)^2, \tag{4}$$

where I_i^t is a superpixel in Φ_i of I_i, and L_i^t is the corresponding superpixel in L_i by adding the motion vector of I_i^t to I_i^t. K_i^t is the motion kernel that we need to estimate for I_i^t. θ is the ratio of the exposure time to the frame interval, and is used to determine the kernel length. We increase θ from 0 to 0.5 in increments of 0.05 to find the optimum kernel length for each superpixel.

Since L_i is the latent frame that we need to reconstruct, we find a sharp neighbor frame I_j to approximate it. We set the sharpness threshold to 0.85 in our implementation. A frame or a superpixel is considered sharp if its sharpness is greater than 0.85. In a video sequence, however, moving objects or camera

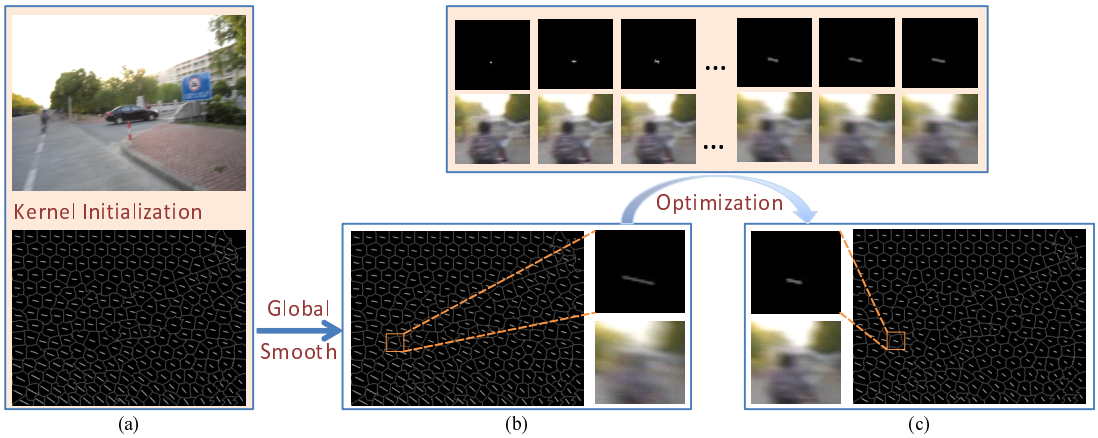

Fig. 3. Estimation of blur kernels. (a) shows blur kernels generated using the original motion field, whereas (b) shows the smoothened blur kernels generated after global optimization. (b) and (c) show the estimation process where we vary parameter θ from 0 to 0.5 in Eq. 3 to optimize the final blur kernels.

shake may cause significant misalignment errors between I_i and I_j. If we let c be the motion vector between I_i^t and I_j^t, where $I_j^t \in \Phi_j$ is the corresponding superpixel of I_i^t, we may define f as an alignment function using c to compensate for the motion; thus, $f(I_j^t)$ approximates I_i^t. In addition, since the alignment error may increase significantly as the frame distance increases, not all video frames with sharpness greater than 0.85 can be candidates to estimate the motion model for I_i. Hence, we set a temporal window:

$$W_i = \{k : \forall j, \max(j-\omega, 0) < k < \min(j+\omega, n-1)\}, \quad (5)$$

where ω is the window size and is set to 5 in our implementation. The motion blur kernel for I_i is then estimated as:

$$E_{blur} = \min_{\theta} \sum_{\substack{j \in W_i \\ \alpha_j > 0.85}} \sum_{\substack{I_i^t \in \Phi_i \\ I_j^t \in \Phi_j}} (\|f(I_j^t) \otimes K_i^t(\theta) - I_i^t\|)^2, \quad (6)$$

Fig. 3 shows the spatially varying blur kernel estimation process, by finding a proper kernel length from the motion field. Since the estimated blur kernels may contain errors due to the inaccuracy of the motion field and the local energy minimization function, we use the blur kernels of neighbor superpixels to optimize the blur kernel for I_i^t. We aim to find an optimal λ to smooth the kernels estimated using the original motion field as:

$$E_{smo} = \min \sum_{I_i^t \in \Phi_i} \left(\|\overline{K_i^t} - K_i^t\|^2 + \lambda \sum_{I_i^n \in N(I_i^t)} (\|\overline{K_i^t} - K_i^n\|^2) \right), \quad (7)$$

where $\overline{K_i^t}$ is the globally optimized blur kernel of K_i^t for I_i^t by minimizing E_{smo}. $N(I_i^t)$ is the set of adjacent superpixels to I_i^t and λ is a smoothing factor to determine the influence of neighbor kernels K_i^n. $\|\cdot\|$ denotes the L2-norm of two blur kernels computed by the motion vectors. $\|\overline{K_i^t} - K_i^t\|$ can be denoted as $\|\overline{\mathbf{R}_i^t} - \mathbf{R}_i^t\|$, where $\mathbf{R} = (u^0, v^0, u^1, v^1)$ is the motion field of K. Thus, Eq. 7 can be optimized as:

$$(E + \lambda G)\overline{\mathbf{R}_i^t} = \mathbf{R}_i^t, \quad (8)$$

where E is the identity matrix and G is the Laplacian matrix calculated by a graph with each superpixel as a node. In our implementation, we set $\lambda = 0.5$. While we should perform the optimization in every kernel estimation step in Eq. 6, the optimization has a high computational cost and may not improve the performance. Hence, we optimize the kernels after we have obtained the motion field of a frame. During the kernel estimation step, we simply use the proper θ value to reduce the alignment error between $f(I_j^t)$ and I_i^t. Fig. 4 shows an example of a blur frame caused by camera shake and a car moving from right to left. Fig. 4(b) shows that homography-based methods [3] [7] fail to approximate the car motion, as they use a homography matrix with 8 degrees of freedom to build the motion model. On the other hand, Fig. 4(c) shows that our method can approximate motion blur caused by camera shake and moving objects well.

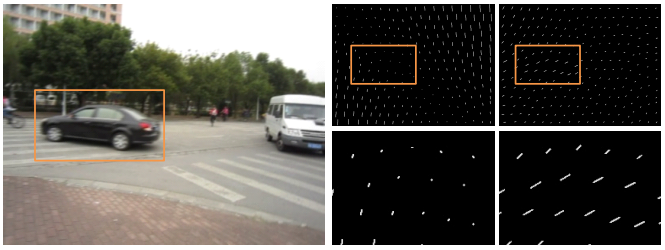

Fig. 4. Comparing our blur kernels with the homography-based blur kernels [3] [7]. (a) shows a blurry frame. (b) shows the homography-based blur kernels that mainly model the camera shake. (c) shows our blur kernels that can model the camera shake as well as the moving car on the left.

D. PatchMatch-based Frame Deblurring

PatchMatch [10] and its generalized algorithm [11] are known to be efficient in finding similar patches for image completion, retargeting and reshuffling. However, the pixel-wise global search is not effective for our superpixel-based search problem. The PatchMatch Filter Algorithm (PMF) [31] involves a search strategy based on superpixels. It first builds an adjacency graph for the input image, where each superpixel is a graph node and an edge connects two adjacent superpixels.

Similar to PatchMatch, the PMF iterates two search strategies, i.e., neighborhood propagation and random search. However, PMF improves the efficiency by replacing the pixelwise search with superpixelwise search. It does not consider spatial and temporal coherences nor motion blur. To address these limitations, we present our PatchMatch-based search strategy based on the original PatchMatch algorithm [10] to search for a sharp superpixel to replace a blurry one using our estimated motion model. Given a superpixel, we adapt the PatchMatch algorithm to search for an irregular-shaped region, rather than a fixed squared region, based on a new distance function that considers color, spatial and temporal constraints, as shown in Eq. 9 to 12. After the PatchMatch initialization step, we iterate through the propagation step and the random search step. However, instead of performing the random search step on the whole frames, we confine the search to a small region to save time. Our argument is that most of the best matching superpixels are near to the original superpixels, as shown in Fig. 5. We elaborate this idea as follows.

To find the most adequate sharp superpixel and maintain spatial coherence, we propose a distance function for two superpixels, instead of SSD (sum of squared differences) in the original PatchMatch algorithm, as:

$$E_{data} = E_{color} + E_{temporal} + E_{spatial}, \tag{9}$$

where E_{color} is the distance between two patches. $E_{temporal}$ and $E_{spatial}$ enforce temporal and spatial coherences in patch matching.

Given superpixel I_i^t, we find the nearest-neighbor field from candidate frame $I_j : j \in W_i$ by the specified rotation angle, scale size, and the length of the blur kernel. $I_j^t(x, y, r, s, k)$ represents the transformation result with offset values along the x-axis, the y-axis, rotation, scale. k represents the length of the blur kernel. It is applied to the pixel of the transformation result obtained by the generalized PatchMatch algorithm with search space (x, y, r, s). Here, we add the motion vector of I_i^t to the superpixel to get the corresponding superpixel (of the same shape), I_j^t, in I_j. The color term E_{color} is then expressed as:

$$E_{color} = \|I_i^t - I_j^t(x, y, r, s, k)\|^2. \tag{10}$$

We assume that the motion field can approximately represent the location of the sharp superpixel. Thus, we apply the motion field constraint between blurry superpixel I_i^t and sharp superpixel I_j^t, and define the temporal term as:

$$E_{temporal} = \sqrt{(x^t - u_{i,j}^t)^2 + (y^t - v_{i,j}^t)^2}, \tag{11}$$

where (x^t, y^t) is the offset value of superpixel I_i^t along x-axis and the y-axis. $(u_{i,j}^t, v_{i,j}^t)$ is the motion vector from I_i^t to I_j^t. Therefore, we minimize the temporal error to maintain temporal coherence when searching for a sharp superpixel.

We also consider the neighbor superpixels and use their motion vectors to find a sharp superpixel, which is similar to the spatial smoothness regularization term [32]. The idea is that I_i^t and its neighbor superpixels should have similar motion vectors, due to spatial coherence. Thus, we apply the motion field constraint between candidate superpixel I_i^t and its neighbor superpixels $I_i^n \in N(I_i^t)$. Our spatial term can be optimized as:

$$E_{spatial} = \sum_{n \in N(I_i^t)} \sqrt{(x^t - u_{i,j}^n)^2 + (y^t - v_{i,j}^n)^2}, \tag{12}$$

Like the motion model estimation step in Section III-C, we use the motion vector to determine the initial search position of all neighbor frames in the PatchMatch initialization step. The initial values for rotation and scale are set to zero. We compute an initial blur kernel for each superpixel in Section III-C. This initialization step can help find proper superpixels within a few iterations.

A random search for superpixel I_i^t is performed during each PatchMatch iteration. The original algorithm [11] randomly assigns locations to all possible positions. Since our experiments in Fig. 5 show that most superpixels can be found in a limited local window with a small range of rotation and scale values, we use a $w \times w$ spatial window centered at the initial pixel position in each superpixel as the search window. In our implementation, we set $w = 15$, which means that the search window for each pixel ranged from -7 to 7. We set the rotation angle from $-\frac{1}{20}\pi$ to $\frac{1}{20}\pi$ and the scale size from 0.8 to 1.25. for the blur kernels, we vary the length ratio of the initial blur kernels from 0.8 to 1.2. We can then obtain the rotation angle, scale size, blur kernel, and position from neighbor sharp frames by minimizing E_{data}(for all $j \in W_i$ and $\alpha > 0.8, \alpha^t > 0.85$) for each blurry superpixel.

Simply replacing the blurry superpixels with the matched sharp superpixels inevitably introduces noise or seams, especially around the boundaries of a moving object. Traditional methods [33] solve this problem based on blending. They sample their patches densely to ensure that neighbor patches overlap with one another, and calculate a weighted average of the overlapping pixels. However, such blending introduces blending blur, which is the artifact that we want to remove. To address this problem, we apply texture synthesis to combine the overlapping shape superpixels at each pixel, using their spatial distances as weights. Suppose that $I_{i,p}^t$ denotes a pixel p in superpixel I_i^t, and $I_i^n \in (I_i^t)$ is a neighbor superpixel. We assume that the minimum spatial Euclidean distance between $I_{i,p}^t$ and I_i^n is d_n. We sample the pixels of all neighbor superpixels and reconstruct the new pixel as:

$$L_{i,p}^t = \frac{\sum_{n \in N(t)} (e^{-0.1 d_n} I_{i,p}^n) + e^{-0.1 d_t} I_{i,p}^t}{\sum_{n \in N(t)} e^{-0.1 d_n} + e^{-0.1 d_t}}, \tag{13}$$

where $I_{i,p}^n$ is the pixel on the enlarged region of superpixel I_i^n at p. Thus, p is in superpixel I_i^t and in the enlarged region of I_i^n. We calculate the nearest spatial distance d_t from p to I_i^t, and d_n from p to I_i^n, as shown in Fig. 6. The final blended pixel is a weighted average of superpixel I_i^t and all its neighbor superpixels on pixel p. We set the spatial distance d_t to 0, as p is on I_i^t. We only consider the neighbor superpixels close to p and set $d_n = \infty$ if d_n is larger than $\frac{1}{2}w$. This is because the larger d_n is, the greater the difference between the value of I_i^t and that of the enlarged region of I_i^n if superpixels I_i^t and I_i^n are obtain from different sharp frames. This approach produces smooth blended results and avoids blending blur.

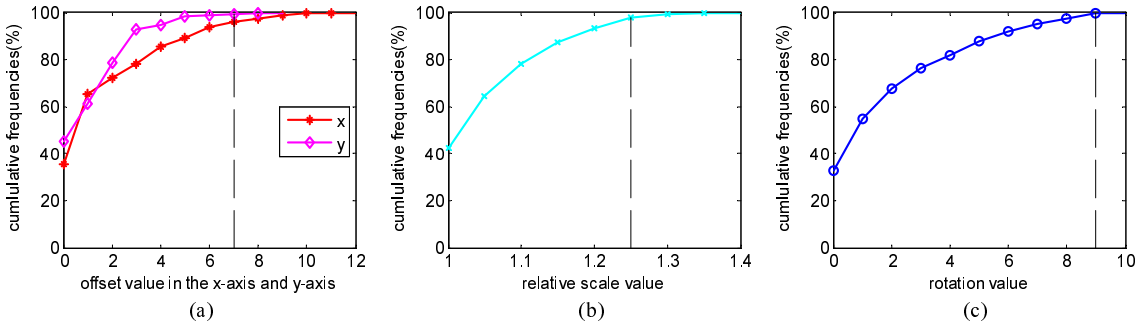

Fig. 5. The best matching nearest-neighbor field found in all frames of a video sequence. (a) shows the offsets in the x-axis and y-axis, (b) shows the best scale size. (c) shows the best rotation angle. The dotted line in each diagram is chosen as the search window in our implementation. Note that most of the best matching superpixels can be found within this search window.

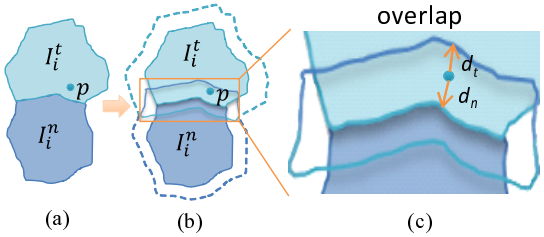

Fig. 6. Blending strategies. (a) shows two adjacent superpixels. (b) shows the enlargement of the adjacent superpixels to overlap each other. (c) shows the blending strategy of a pixel by computing the distances to the two superpixels.

Once a blurry superpixel is sharpened, we update its sharpness value. As a result, the sharpness of the frame may also increase. If it is above the sharpness threshold, it may then be used to reconstruct other blurry frames.

IV. Implementation Issues

To speed up the proposed deblurring algorithm, we considered the following issues.

A. Spreading of Sharp Superpixels

As the order of deblurring video frames may affect processing time, Cho *et al.* [7] process the sharpest frames first. While this quickly produces frames with high sharpness values, it may not always be easy to find sharp superpixels if sharp frames appear only sparsely in the video.

To quickly spread sharp superpixels, we order the video frames by the number of sharp neighbor frames that they have. The deblurring algorithm will be performed first on the one with the highest number of neighbor frames in its temporal window. After a frame has been deblurred, its neighbor frames may now have a different number of sharp neighbor frames and we update the sorted list to reflect this situation. This approach is to ensure that we process the frames that are more likely to find good matches first.

B. GPU-based Acceleration

Since all deblurring steps are performed on individual superpixels, they can be easily parallelized to run on GPUs. Except for the optical flow estimation step [29], we have implemented all other steps based on OpenGL Shading Language (GLSL).

Segmentation. The original SLIC requires several iterations to find the cluster centers and to assign a cluster label for each pixel. This is difficult to perform in parallel on GPUs, as this assignment step is done iteratively based on the cluster centers. To address this problem, we treat each pixel as an independent unit and try to find a cluster center for it, instead of finding what pixels belong to a cluster. In addition, due to temporal coherence, we can transfer the cluster centers and labels in one frame as input to the next to improve efficiency.

Kernel estimation and patch matching. As these steps are applied to each superpixel, they can be performed after the segmentation step. However, the global smoothing step (Eq. 8) needs to solve a linear system and thus run on the CPU, before assigning tasks to GPUs. We compute all values of θ, i.e., from 0 to 0.5 with increments of 0.05, in the GPU to find the proper kernel length for each superpixel, which is used as the initial value for superpixel matching. In superpixel matching, we apply the jump flood scheme [34] with 4 neighbors in the propagation step, as in [10], on a spatial window of $w \times w$. In the random searching step of patch matching, we need to generate random numbers. However, as GLSL does not provide a random number generator, we use Poisson noise to synthesize random numbers.

V. Experimental Results

We have implemented the proposed method in C++ and tested it on a PC with an i3 3.4 GHz CPU. Using only a single thread, it takes about 10 minutes to deblur a 800×600 video frame. We have also tested it on the same PC with a GeForce GTX 750Ti graphics card. The computation time reduces to 20 seconds. In this section, we evaluate the proposed method in detail.

A. Deblurring Model Testing

We first compare the proposed motion model with the homography-based model [3] [7]. Homography-based motion models may produce good deblurred results for blurs caused by camera motions. As shown in Fig. 4(b), it fails to estimate object motion, i.e., the moving car. Fig. 7 compares the two

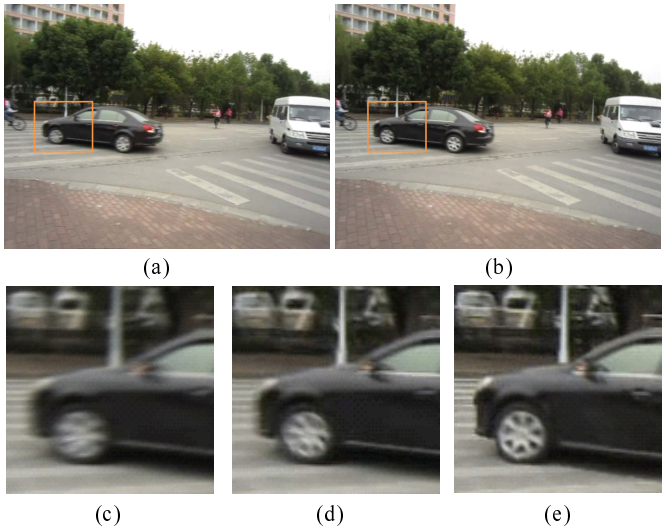

Fig. 7. Comparing the homography-based model with the proposed model as in Fig. 4. (a) Deblurring result using homography-based blur kernels similar to [7]. (b) Deblurring result using our blur model. (c)-(d) are magnified views of the input image, of (a) and of (b), respectively.

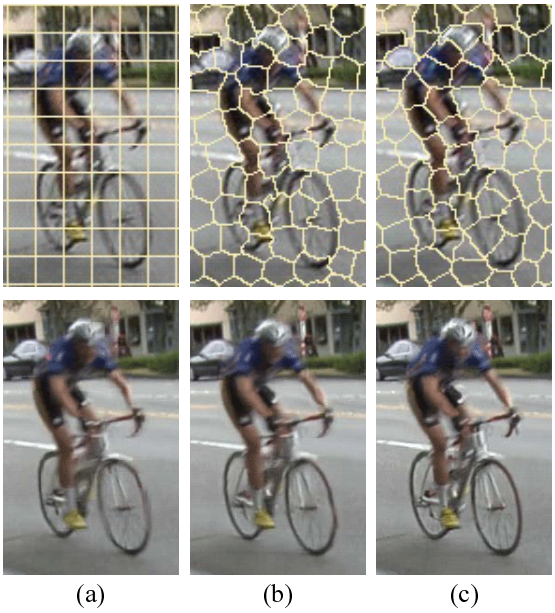

Fig. 8. Comparing different image segmentation approaches for deblurring. (a) Deblurring result using regular patch-based representation. (b) Deblurring result using SLIC superpixels. (c) Deblurring result using our motion-based SLIC superpixels. Note that the front wheel is much smoother.

results. Although the motion blur appearing at the background, caused by camera shake, are sharpened in both Fig. 7(c) and 7(d), the homography-based method fails to recover the motion of the car correctly. Hence, it also fails to find suitable patches to deblur the moving car. On the contrary, our result, which is obtained using a spatial varying motion model, is more convincing.

We have also studied the effects of using different segmentation methods in our deblurring model. As regular patch-based representation divides each image regularly, it does not handle

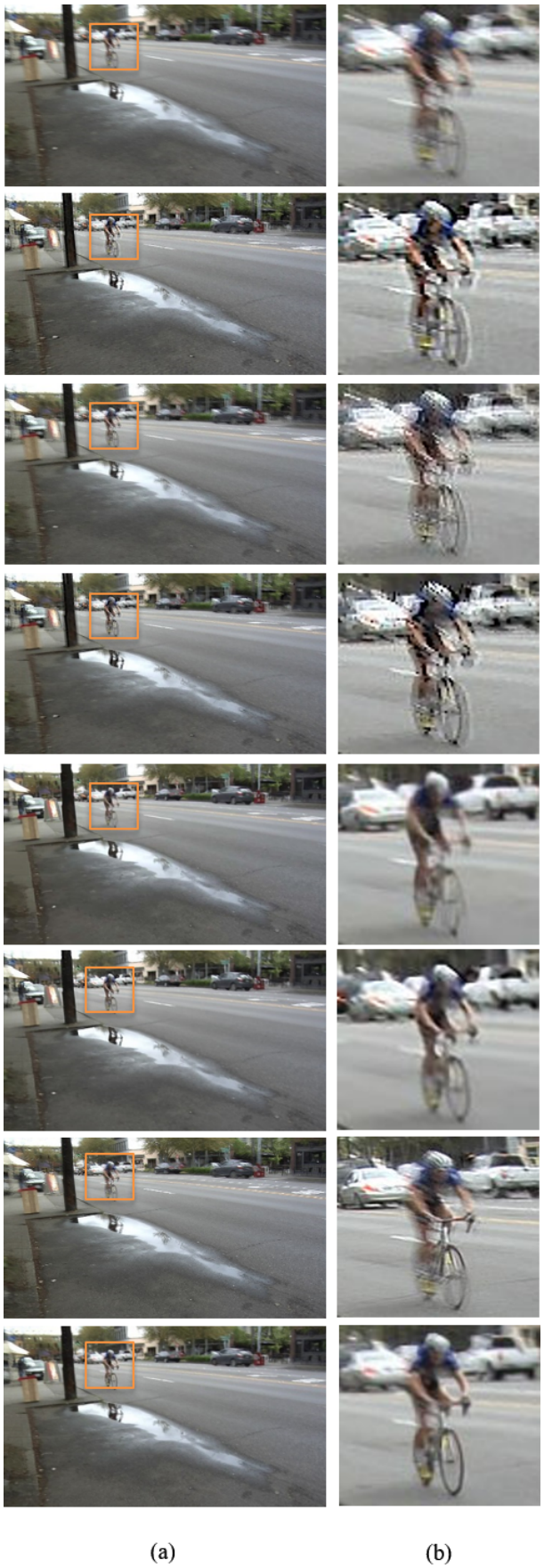

Fig. 9. Comparing different deblurring approaches on a bicycle video frame. From top to bottom: input frame, followed by results of [12], [14] [15], [7], [28], [27] and our method.

object segmentation well, and may fail in motion estimation and in patch matching along object boundaries as shown in Fig. 8(a). The superpixel-based segmentation methods shown in Fig. 8(b) and 8(c) produce better segmentation, which helps improve the accuracy of superpixel matching. On the other hand, as our method considers object motion, it produces more consistent segmentation across frames. Hence, our reconstructed frame preserves the object structure, e.g., the front wheel, better.

B. Visual Evaluation

We compare the proposed method with differents methods, including single-image deblurring methods using a uniform blur kernel [12] [35] and using a non-uniform blur kernel [14] [15], video deblurring method [7] and two state-of-art method [28] [27]. As we do not have the code for [7] and [27], we can only compare with their results in Fig. 9 and 10.

From Fig. 9 and 10, we observe that the results from [12] contain obvious artifacts, as it applies a uniform blur kernel to restore the frames. Although a non-uniform blur kernel is able to handle camera shake, the results from [14] [15] have ringing artifacts. The complex motion blur increases the difficulty of kernel estimation. The image background from [7] is much better than those from single-image deblurring methods. Like ours, [7] is a patch-based video deblurring method. However, it cannot handle moving objects and may fail to find proper sharp patches for moving objects using the homography-based motion model. Thus, the blended result of these mismatched patches may not show any improvement or may sometimes be worse than the original (see the car in Fig. 10(g)). In addition, as it segments the image regularly, it increases the chance of getting mismatches along the moving object boundaries, which may destroy the structure of the moving object, as shown in the fifth row of Fig. 9(b). Although the shock filter is used to sharpen the patches, the frame quality is only slightly improved. The proposed motion deblurring model uses appropriate blur kernels to blur the sharp superpixels found before superpixel matching. This helps find suitable sharp superpixels more accurately, as shown in Fig. 10(j). Comparing [28] with the proposed method, our method shows shaper results. Although [27] produces better background details, the video still exhibits motion blur. Further, in Fig. 10, while the tire is clearer, the structure is wrong.

In Fig. 11, we compare the proposed method with state-of-the-art single-image deblurring methods, which use uniform [12] and non-uniform [14] [15] motion models, and the pixel-wise deblurring method [28]. While deconvolution with a uniform blur kernel may improve the quality of the image background to a certain extent, it introduces severe ringing artifacts on the moving objects, as shown in the second row of Fig. 11. Although non-uniform motion models [14] [15] may produce fewer ringing artifacts, it is very difficult to recover accurate blur kernels from a single image for deconvolution. The proposed method produces better result by avoiding this deconvolution step.

Fig. 12, 13 and 14 show video frames involving significant blurs caused by fast moving objects, while the blackgrounds

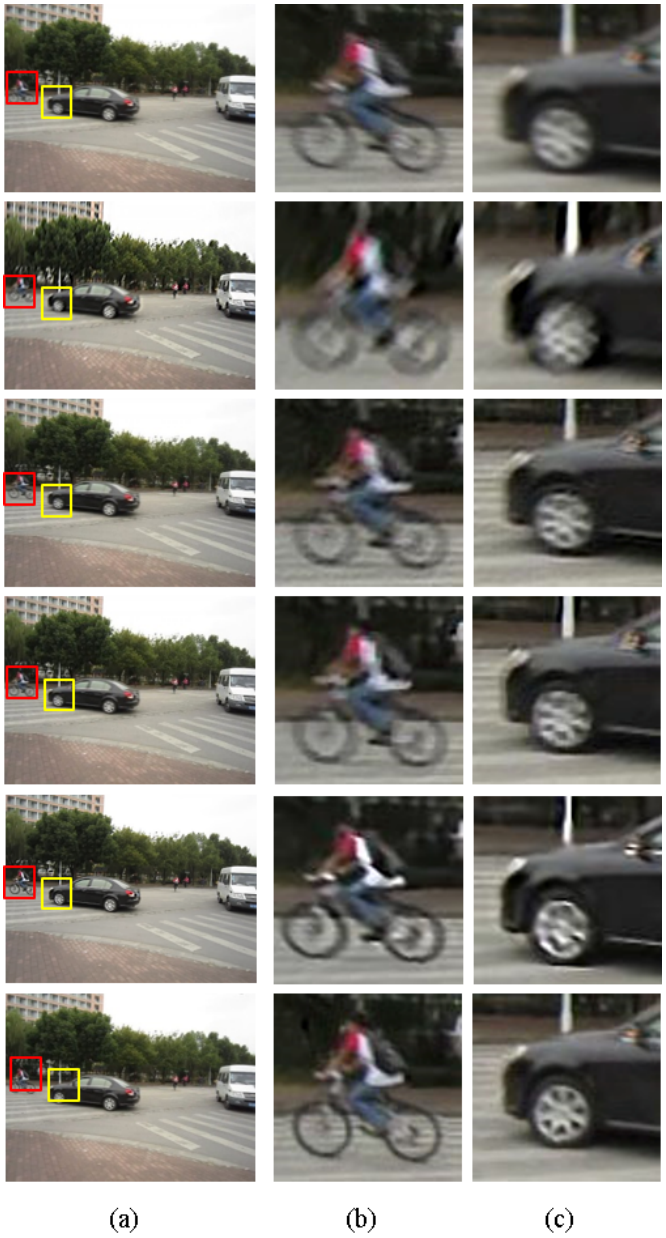

Fig. 11. Deblurring a car-bicycle frame. From top to bottom: the input frame, followed by results of [12], [14], [15] [28] and our method.

are blurred in different degrees. In Fig. 12, most methods do not perform very well. Our method also fails to deblur the cyclist/bicycle because they are blurred throughout the video. However, it can recover the building, in particular the window, much better than the other methods. In Fig. 13, all other methods produce obvious artifacts in the car wheels, while our performs very well in particular the text on the car. However, our method fails to deblur the background as the background is blurred throughout the video clip. In Fig. 14, our method can deblur both the foreground cyclist (the text on the body in particular) and the background very well, compared to all the other methods.

Images or videos captured by image sensors may be noisy when filmed under a dim environment. Most deconvolution-

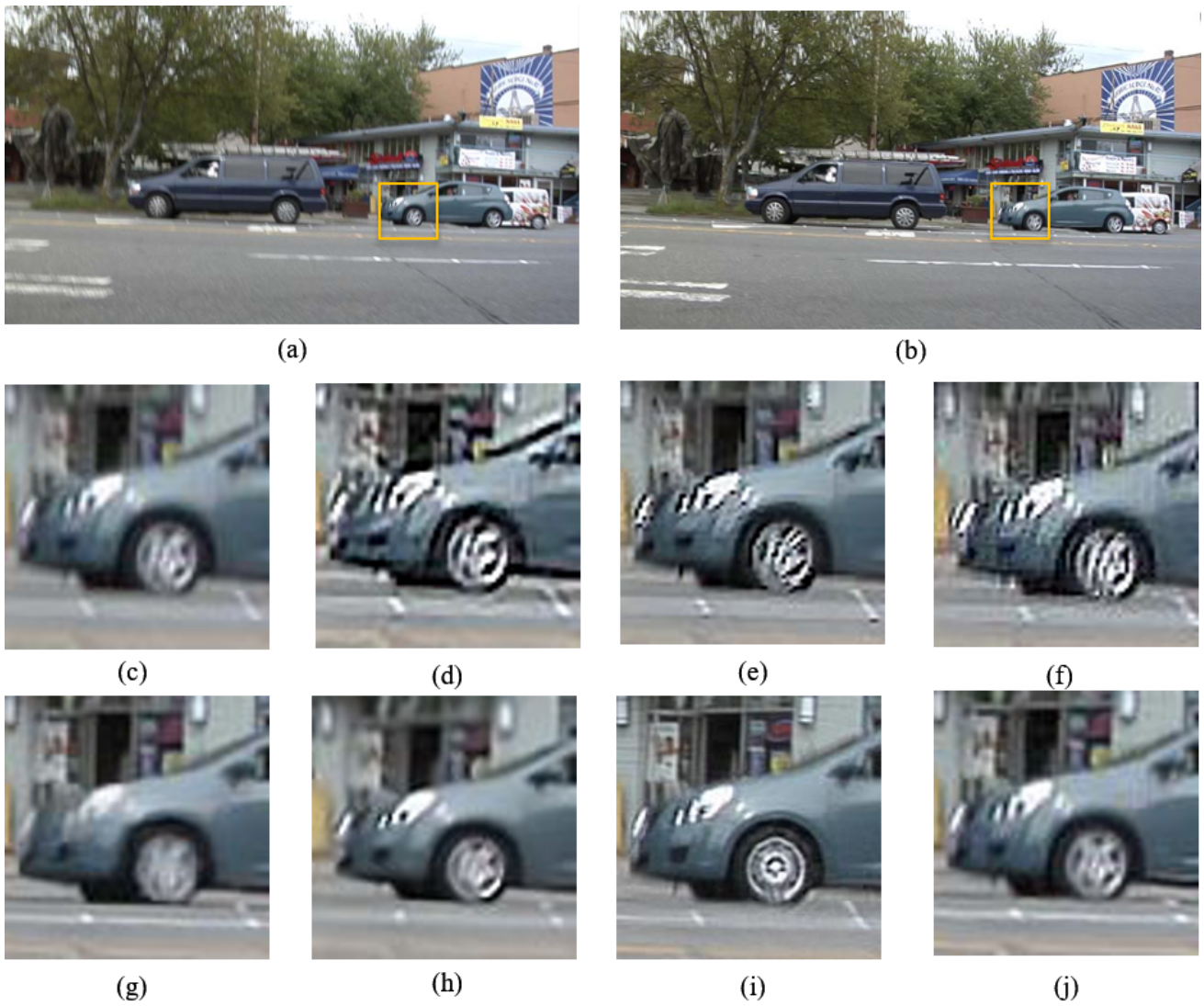

Fig. 10. Comparing different deblurring approaches on a car video frame. (a) Input frame. (b) Deblurred frame using our method, (c) to (j) show the magnified views of the input frame, followed by results of [12], [14], [15], [7] [28] [27] and our method. Note that although the background of (g) is sharper, the moving car is actually more blur than (a), since [7] is unable to correctly handle moving objects.

Fig. 12. Deblurring of a bicycle frame. (a) Input frame. (b) Magnified view of the input frame. (c) Result of [12]. (d) Result of [15]. (e) Result of [28]. (f) Result of our method.

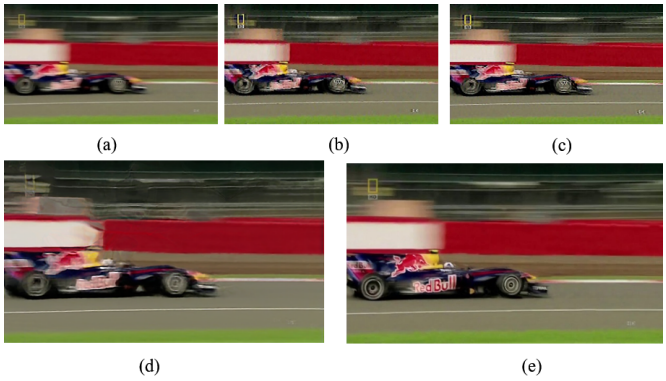

Fig. 13. Deblurring a fast car frame. (a) Input frame. (b) Result of [12]. (c) Result of [15]. (d) Result of [28]. (e) Result of our method.

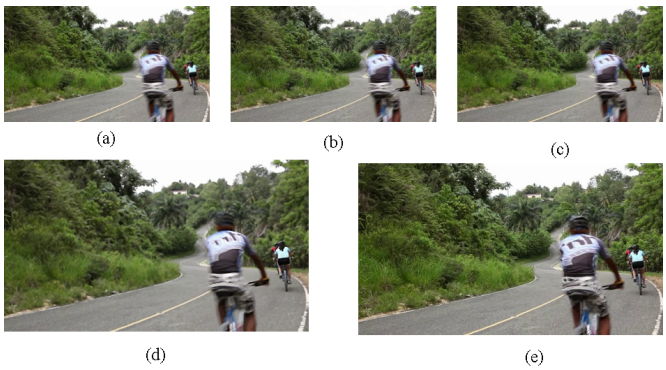

Fig. 14. Deblurring another bicycle frame. (a) Input frame. (b) Result of [12]. (c) Result of [15]. (d) Result of [28]. (e) Result of our method.

TABLE I
QUALITY ASSESSMENT OF RESULTS IN FIGS. 9 AND 10, USING NIQE [36] AND BRISQUE [37]. (LARGER VALUES INDICATE LOWER QUALITY, AND NUMBERS IN BOLD INDICATE THE BEST RESULTS)

Video	IQA	Input	[7]	[28]	[27]	Ours
Bicycle (Fig. 9)	NIQE	5.626	3.585	4.488	4.065	**3.208**
	BRISQUE	34.566	33.855	38.000	21.376	**19.650**
Car (Fig. 10)	NIQE	5.148	3.399	5.063	3.839	**3.221**
	BRISQUE	39.134	36.297	42.013	**27.298**	34.751

TABLE II
QUALITY ASSESSMENT OF RESULTS IN FIG. 11 — 16, USING NIQE [36] AND BRISQUE [37]. (LARGER VALUES INDICATE LOWER QUALITY, AND NUMBERS IN BOLD INDICATE THE BEST RESULTS

Video	IQA	Input	[12]	[15]	[28]	Ours
Fig. 11	NIQE	3.716	4.742	3.227	4.529	**3.089**
	BRISQUE	32.732	49.412	27.221	37.407	**25.516**
Fig. 12	NIQE	4.705	3.653	3.844	5.390	**3.530**
	BRISQUE	32.550	22.258	25.704	**21.812**	29.679
Fig. 13	NIQE	5.251	**4.425**	4.9241	5.507	4.451
	BRISQUE	41.290	**29.679**	32.314	48.834	30.508
Fig. 14	NIQE	3.843	4.734	5.617	**3.0641**	3.806
	BRISQUE	43.527	49.178	57.057	**29.845**	43.376
Fig. 15	NIQE	4.509	3.854	3.899	6.184	**3.763**
	BRISQUE	35.508	33.590	34.069	38.698	**32.783**
Fig. 16	NIQE	5.871	5.109	4.712	5.378	**3.980**
	BRISQUE	41.094	38.486	37.184	**30.488**	34.268

based image deblurring methods are sensitive to image noise. As our method is deconvolution-free, it can avoid such problems. Fig. 15 and 16 demonstrate this feature. Although [35] handle noise in deblurring and [12] is robust to noise, they produce ringing artifacts. Fig. 16 is also noisy. As a result, their outputs may appear worse than the inputs. The proposed method, on the other hand, is able to deblur the input frames in both situations.

C. Quality Assessment Evaluation

We evaluate the deblured results from [12], [15], [7] [28] [27] and the proposed method using two no-reference image quality assessment methods, NIQE [36] and BRISQUE [37]. These two methods perform better than most latest methods and are correlated with human perception [38]. NIQE mainly extracts highly sharp part of an image by a spatial domain natural scene statistic model, which can evaluate the image quality without training. On the contrary, BRISQUE is a machine learning based approach independent of database content. It uses the statistics of normalized local luminance coefficients under natural scenes and can evaluate image quality holistically without requiring that the user supply specific features such as ringing, blur or blocking in advance.

Table I shows the average image quality of the deblurring results. Our method outperforms the other three video deblurring methods [7] [28] [27] on the first three rows. [27] outperforms ours in the last row. Table II shows that our method performs mostly better than the image deblurring methods [12] and [15] on both image quality assessment metrics. It also outperforms the other methods on six test cases, while [28] outperforms other methods on four test cases. Although image deconvolution helps improve image quality, it introduces the ringing artifact, which affects the performances of [12] and [15] on quality assessment metrics. However, for Fig. 12, [12] and [15] can restore sharp silhouettes with few ringing artifacts. On the other hand, our method can only reconstruct a frame to one as sharp as other frames in the video can provide. As this entire video is blurred, our method performs worse than [12] and [15] on BRISQUE. For Fig. 13, while our method is able to reconstruct the fast car, the background is nearly unchanged. On the other hand, [12] produces a slightly sharper background. Hence, it yielded a slightly better performance on both NIQE and BRISQUE. It is interesting to note that [12] and [15] perform even worse than the input video on Fig. 14. This is because the background in this video was already sharp enough. Applying image deconvolution to these sharp regions lead to ringing artifacts, which affect the performance of these methods on the two metrics.

D. Computation Time Evaluation

We have compared the average computation time per frame of each video sequence using different deblurring methods,

TABLE III
AVERAGE COMPUTATION TIME PER FRAME (SEC) FOR DIFFERENT METHODS. THE RIGHT HALF SHOWS VARIATIONS OF OUR METHOD.

Video	[12]	[35]	[14]	[15]	CPU-based	GPU-based	GPU-based (with [7] to order frames)
Fig. 9 (640×360)	1.89	12.30	303	1370	458	18.06	45.03
Fig. 10 (640×360)	1.82	12.43	143	1234	437	18.04	36.57
Fig. 11 (640×480)	2.56	17.36	150	1298	466	19.12	46.61
Fig. 12 (640×480)	2.45	18.05	493	1370	518	19.89	37.31
Fig. 13 (640×360)	2.67	12.79	288	1215	429	16.71	49.38
Fig. 14 (640×360)	2.34	12.57	259	1217	425	16.40	41.14
Fig. 15 (640×480)	2.48	17.06	176	1308	474	18.23	49.34
Fig. 16 (640×480)	2.40	17.24	198	1289	454	17.81	46.58

as shown in Table III. [12] is the most efficient method, as it only estimates a global blur kernel for the input image. Although a uniform blur kernel is estimated for deconvolution, [35] needs to apply a series of directional filters to refine the blur kernel, resulting in longer computation time. Both [15] and [14] require higher computation times in order to estimate non-uniform blur kernels. The CPU implementation of our method is also slow. However, the GPU implementation of our method has a speed up of roughly 25 times. We have further compared the computational times of the GPU implementation by considering the speedup technique discussed in Section IV-A. As shown in the last column of Table III, it takes at least double the time to process a video in most situations when using the frame ordering technique proposed in [7].

VI. CONCLUSION

In this paper, we have proposed a structure-preserving video deblurring algorithm that uses irregular motion-based segmentation to estimate a non-uniform motion model and perform superpixel deblurring. The proposed method assumes that sharp frames or superpixels are sparsely spread in video sequence and uses sharp superpixels to reconstruct blurry ones. The superpixels are essential for effectively distinguishing moving objects from the background and are more robust than regular patches employed by previous methods. Experimental results demonstrate that the proposed method can reconstruct blurry frames on both static and moving objects while maintaining spatial and temporal coherence without introducing artifacts.

In general, our method fails if no sharp superpixels can be found from other frames of the video. In future work, we intend to address this limitation. One possible approach that we are looking at is to apply deconvolution on frames to recover blur kernels more reliably to form the initial sharp frames.

ACKNOWLEDGMENTS

The authors would like to thank all reviewers for their helpful suggestions and constructive comments. The work is supported by the National Natural Science Foundation of China (No. 61632003, 61572316, 61672502, and 61671290), National High-tech R&D Program of China (863 Program) (No. 2015AA015904), the Key Program for International S&T Cooperation Project (No. 2016YFE0129500) of China, the Science and Technology Commission of Shanghai Municipality (No. 16DZ0501100), a SRG grant from City University of Hong Kong (No. 7004415), Natural Science Foundation of Jiangsu Province (No. BK20140404), and the interdisciplinary Program of Shanghai Jiao Tong University (No. 14JCY10).

REFERENCES

[1] Y. Tai, H. Du, M. Brown, and S. Lin, "Image/video deblurring using a hybrid camera," in *Proc. IEEE CVPR*, 2008.
[2] H. Zhang, D. Wipf, and Y. Zhang, "Multi-image blind deblurring using a coupled adaptive sparse prior," in *Proc. IEEE CVPR*, 2013.
[3] Y. Li, S. Kang, N. Joshi, S. Seitz, and D. Huttenlocher, "Generating sharp panoramas from motion-blurred videos," in *Proc. IEEE CVPR*, 2010.
[4] M. Onogi and H. Saito, "Mosaicing and restoration from blurred image sequence taken with moving camera," in *Proc. ICAPR*, 2005.
[5] S. Cho, Y. Matsushita, and S. Lee, "Removing non-uniform motion blur from images," in *Proc. IEEE ICCV*, 2007.
[6] D. Lee, S. Jeong, Y. Lee, and B. Song, "Video deblurring algorithm using accurate blur kernel estimation and residual deconvolution based on a blurred-unblurred frame pair," *IEEE TIP*, vol. 22, no. 3, pp. 926–940, Mar. 2013.
[7] S. Cho, J. Wang, and S. Lee, "Video deblurring for hand-held cameras using patch-based synthesis," *ACM TOG*, vol. 31, no. 4, Jul. 2012.
[8] Z. Liu, L. Yuan, X. Tang, M. Uyttendaele, and J. Sun, "Fast burst images denoising," *ACM TOG*, vol. 33, no. 6, 2014.
[9] S. Liu, L. Yuan, P. Tan, and J. Sun, "Bundled camera paths for video stabilization," *ACM TOG*, vol. 32, no. 4, 2013.
[10] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman, "Patchmatch: a randomized correspondence algorithm for structural image editing," *ACM TOG*, vol. 28, no. 3, Aug. 2009.
[11] C. Barnes, E. Shechtman, D. Goldman, and A. Finkelstein, "The generalized patchmatch correspondence algorithm," in *Proc. ECCV*, 2010.
[12] L. Xu and J. Jia, "Two-phase kernel estimation for robust motion deblurring," in *Proc. ECCV*, 2010.
[13] M. Ben-Ezra and S. Nayar, "Motion deblurring using hybrid imaging," in *Proc. IEEE CVPR*, 2003.
[14] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce, "Non-uniform deblurring for shaken images," *IJCV*, vol. 98, no. 2, pp. 168–186, 2012.
[15] L. Xu, S. Zheng, and J. Jia, "Unnatural l0 sparse representation for natural image deblurring," in *Proc. IEEE CVPR*, 2013.
[16] A. Levin, "Blind motion deblurring using image statistics," in *Proc. NIPS*, 2006.
[17] Q. Shan, W. Xiong, and J. Jia, "Rotational motion deblurring of a rigid object from a single image," in *Proc. IEEE ICCV*, 2007.
[18] S. Dai and Y. Wu, "Motion from blur," in *Proc. IEEE CVPR*, 2008.
[19] T. Kim and K. Lee, "Segmentation-free dynamic scene deblurring," in *Proc. IEEE CVPR*, 2014.
[20] A. Rav-Acha and S. Peleg, "Two motion-blurred images are better than one," *Pattern Recognition Letters*, vol. 26, no. 3, pp. 311–317, Feb. 2005.
[21] J. Chen, L. Yuan, C. Tang, and L. Quan, "Robust dual motion deblurring," in *Proc. IEEE CVPR*, 2008.

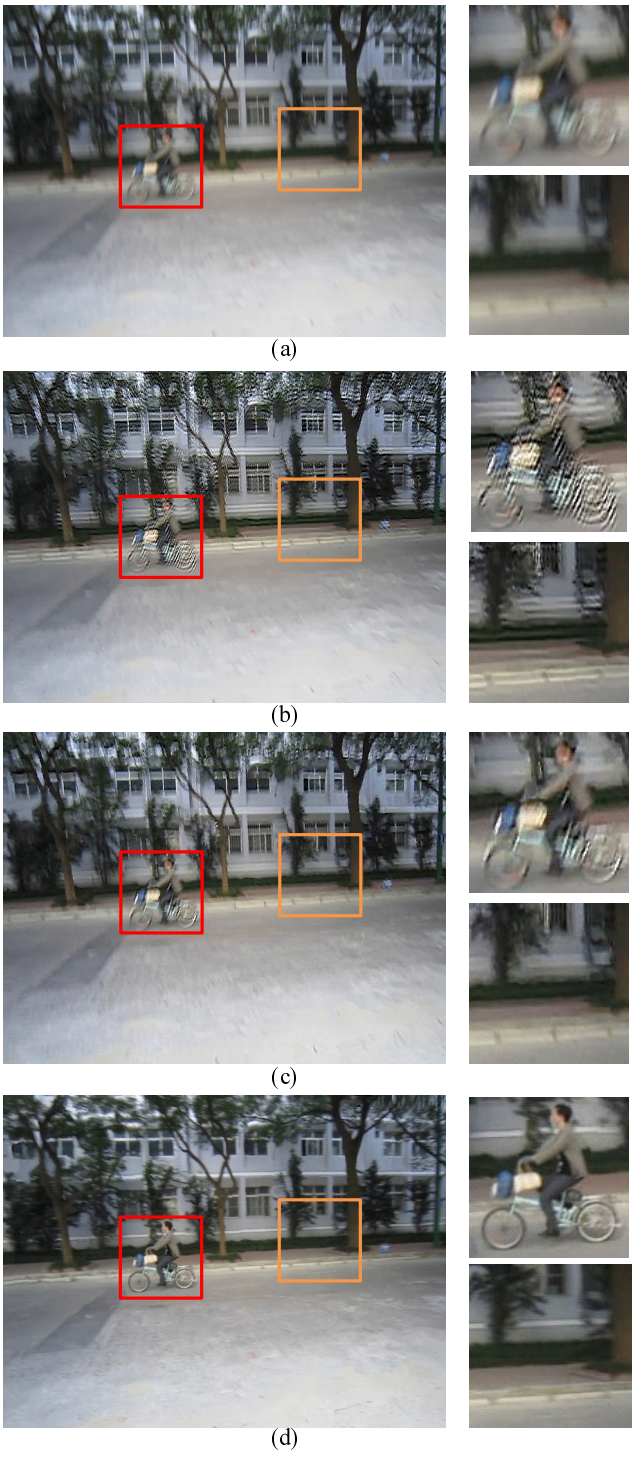

Fig. 15. Deblurring a blurry frame with noise. (a) Input frame. (b) Result from a non-uniform blur kernel with noise handling [35]. (c) Result from [12]. (d) Result from our method.

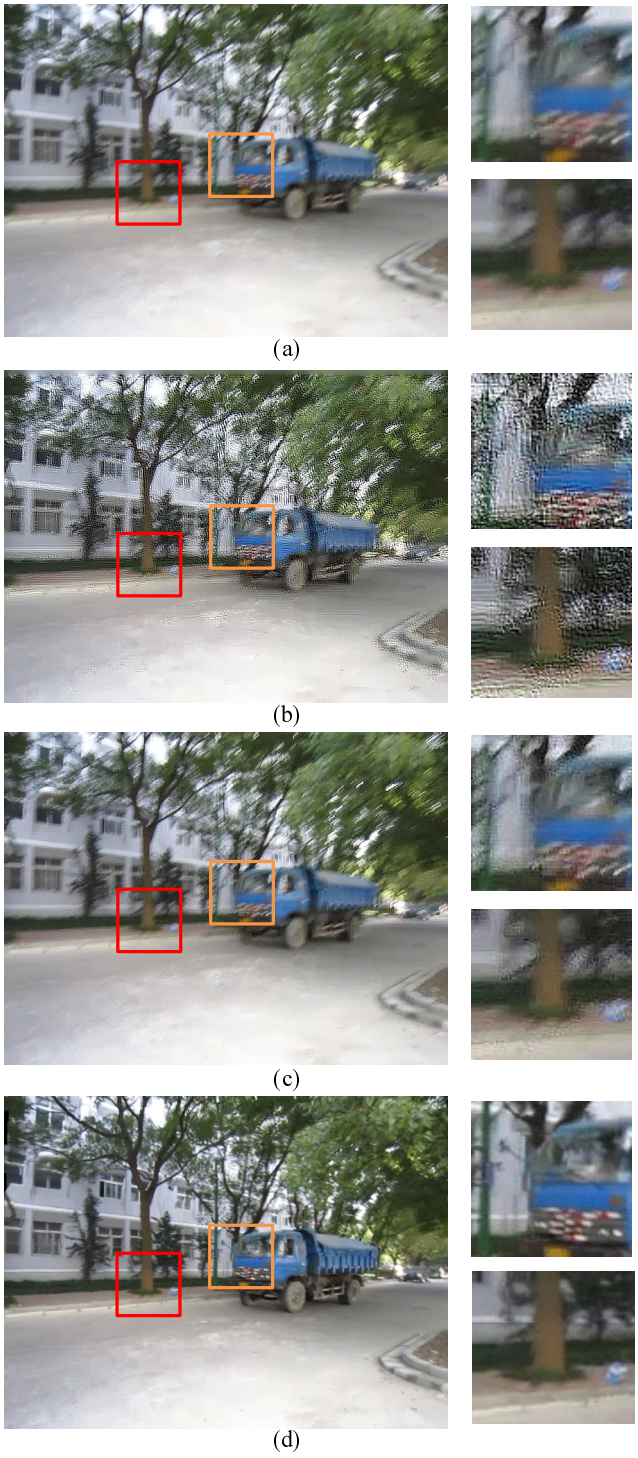

Fig. 16. Deblurring a blurry frame with noise. (a) Input frame. (b) Result from a non-uniform blur kernel with noise handling [35]. (c) Result from [12]. (d) Result from our method.

[22] F. Sroubek and P. Milanfar, "Robust multichannel blind deconvolution via fast alternating minimization," *IEEE TIP*, vol. 21, no. 4, pp. 1687–1700, Apr. 2012.
[23] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H. Shum, "Full-frame video stabilization with motion inpainting," *IEEE TPAMI*, vol. 28, no. 7, pp. 1150–1163, Jul. 2006.
[24] X. He, T. Luo, S. Yuk, and K. Chow, "Motion estimation method for blurred videos and application of deblurring with spatially varying blur kernels," in *Proc. ICCIT*, 2010, pp. 355–359.
[25] H. Takeda and P. Milanfar, "Removing motion blur with spacetime processing," *IEEE TIP*, vol. 20, no. 10, pp. 2990–3000, 2011.
[26] J. Wulff and M. J. Black, "Modeling blurred video with layers," in *Proc. IEEE CVPR*, 2014, pp. 236–252.
[27] M. Delbracio and G. Sapiro, "Hand-held video deblurring via efficient fourier aggregation," *IEEE Trans. on Computational Imaging*, vol. 1, no. 4, pp. 270–283, 2015.
[28] T. Kim and K. Lee, "Generalized video deblurring for dynamic scenes," in *Proc. IEEE CVPR*, 2015.

[29] L. Bao, Q. Yang, and H. Jin, "Fast edge-preserving patchmatch for large displacement optical flow," *IEEE TIP*, pp. 4996–5006, Dec. 2014.
[30] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE TPAMI*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012.
[31] J. Lu, H. Yang, D. Min, and M. Do, "Patch match filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation," in *Proc. IEEE CVPR*, 2013.
[32] T. Portz, L. Zhang, and H. Jiang, "Optical flow in the presence of spatially-varying motion blur," in *Proc. IEEE CVPR*, 2012.
[33] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, "Texture optimization for example-based synthesis," *ACM TOG*, vol. 24, no. 3, 2005.
[34] G. Rong and T. Tan, "Jump flooding in gpu with applications to voronoi diagram and distance transform," in *Proc. ACM I3D*, 2006.
[35] L. Zhong, S. Cho, D. Metaxas, S. Paris, and J. Wang, "Handling noise in single image deblurring using directional filters," in *Proc. IEEE CVPR*, 2013.
[36] A. Mittal, R. Soundararajan, and A. Bovik, "Making a "completely blind" image quality analyzer," *IEEE Signal Processing Letters*, vol. 20, no. 3, pp. 209–212, Mar. 2013.
[37] A. Mittal, A. Moorthy, and A. Bovik, "No-reference image quality assessment in the spatial domain," *IEEE TIP*, vol. 21, no. 12, pp. 4695–4708, Dec. 2012.
[38] A. Nouri, C. Charrier, A. Saadane, and C. Fernandez-Maloigne, "Statistical comparison of no-reference images quality assessment algorithms," in *Proc. IEEE CVCS*, 2013.

Congbin Qiao received his B.Eng degree in Software College at Shandong University. He is now a graduate student in the Department of Computer Science and Engineering at Shanghai Jiao Tong University. His research interests include image/video processing .

Rynson W.H. Lau received his Ph.D. degree from University of Cambridge. He was on the faculty of Durham University and Hong Kong Polytechnic University. He is now with City University of Hong Kong. Rynson serves on the Editorial Board of Computer Graphics Forum, and Computer Animation and Virtual Worlds. He has served as the Guest Editor of a number of journal special issues, including ACM Trans. on Internet Technology, IEEE Multimedia, IEEE Trans. on Multimedia, IEEE Trans. on Visualization and Computer Graphics, and IEEE Computer Graphics and Applications. In addition, he has also served in the committee of a number of conferences, including Program Co-chair of ACM VRST 2004, ACM MTDL 2009, IEEE U-Media 2010, and Conference Co-chair of CASA 2005, ACM VRST 2005, ICWL 2007, ACM MDI 2009, ACM VRST 2010, ACM VRST 2014. His research interests include computer graphics and computer vision.

Bin Sheng received his BA degree in English and BE degree in computer science from Huazhong University of Science and Technology in 2004, and MS degree in software engineering from University of Macau in 2007, and PhD Degree in computer science from The Chinese University of Hong Kong in 2011. He is currently an associate professor in Department of Computer Science and Engineering at Shanghai Jiao Tong University. He is an associate editor of IET Image Processing. His research interests include virtual reality, computer graphics.

Benxuan Zhang received his B.Eng degree in Computer Science at Shanghai Jiao Tong University. He is now a graduate student in the Department of Computer Science and Engineering at Shanghai Jiao Tong University. His research interests include image processing and deep learning .

Enhua Wu received his B.S. degree from Tsinghua University in 1970, and the PhD degree from the University of Manchester (UK) in 1984. He is currently a research professor at the State Key Lab. CS, Chinese Academy of Sciences. He has also been teaching at the University of Macau, where he is currently an Emeritus Professor. He has been invited to chair or co-chair various conferences or program committees including Siggraph Asia 2016, ACM VRST 2010, 2013, 2015, WSCG 2012. He is an associate editor-in-chief of the Journal of Computer Science and Technology, and the editorial board member of TVC, JVI, CAVW, IJIG, IJSI. His research interests include realistic image synthesis, physically based simulation and virtual reality. He is a member of the IEEE and ACM, and a fellow of the China Computer Federation (CCF).