Video Deblurring for Hand-held Cameras Using Patch-based Synthesis *

Sunghyun Cho[1,2] Jue Wang[2] Seungyong Lee[1]
[1]POSTECH [2]Adobe Systems

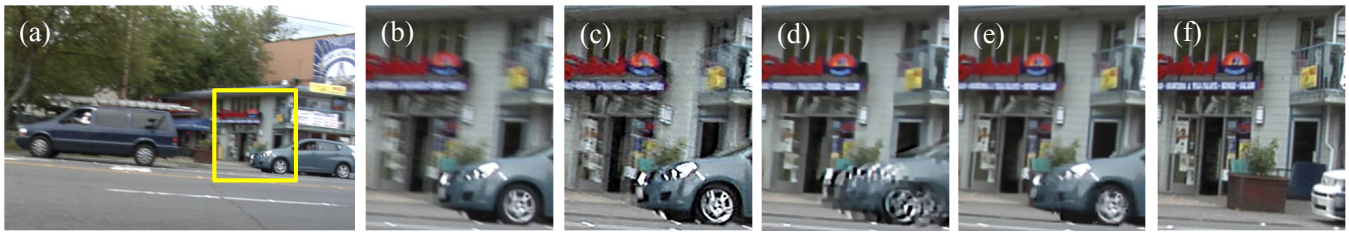

Figure 1: *Comparison of deblurring results of a motion-blurred video frame using different approaches. (a) A blurry frame of the "cars" video. (b) Magnified view of an input region. (c) Single image deblurring. (d) Multi-frame deblurring. (e) Our method. (f) Similar image region from a nearby sharp frame.*

Abstract

Videos captured by hand-held cameras often contain significant camera shake, causing many frames to be blurry. Restoring shaky videos not only requires smoothing the camera motion and stabilizing the content, but also demands removing blur from video frames. However, video blur is hard to remove using existing single or multiple image deblurring techniques, as the blur kernel is both spatially and temporally varying. This paper presents a video deblurring method that can effectively restore sharp frames from blurry ones caused by camera shake. Our method is built upon the observation that due to the nature of camera shake, not all video frames are equally blurry. The same object may appear sharp on some frames while blurry on others. Our method detects sharp regions in the video, and uses them to restore blurry regions of the same content in nearby frames. Our method also ensures that the deblurred frames are both spatially and temporally coherent using patch-based synthesis. Experimental results show that our method can effectively remove complex video blur under the presence of moving objects and other outliers, which cannot be achieved using previous deconvolution-based approaches.

CR Categories: I.4.3 [Image Processing and Computer Vision]: Enhancement—Sharpening and deblurring

Keywords: motion blur, video deblurring, lucky region, patch-based synthesis

Links: DL PDF WEB

1 Introduction

Camera motion is one of the most important factors that differentiate professional videos from ones captured by armature users. Professional videos are often shot using special equipments such as dollies or steadicams to achieve smooth camera motion, while amateur ones are often shot by hand-held cameras with significant camera shake. The impact of a shaky camera on the captured video is twofold: first, it introduces temporal jitter to the video content which is unpleasant to watch; second, it blurs video frames significantly at times when the camera shake is intense.

Video stabilization systems [Liu et al. 2011; Grundmann et al. 2011] have been proposed recently to smooth the camera motion in a shaky video. Although these approaches can successfully stabilize the video content, it leaves the blurriness caused by original camera motion untouched. As a result, the blurry frames become the most noticeable artifact in a stabilized video, as can be seen in the supplementary video.

On the other hand, blurry video frames also hinders video stabilization approaches from achieving high quality results. Most stabilization systems rely on feature tracking to plan the camera motion. However feature tracking over blurry frames is often not reliable due to the lack of sharp image features. Restoring sharp frames from blurry ones caused by camera motion, which we dub *video motion deblurring*, is thus critical to generating high quality stabilization results. For this reason we argue that video deblurring should be done prior to applying stabilization techniques. This is in sharp contrast to the deblurring workflow proposed in previous systems [Matsushita et al. 2006], where simple deblurring techniques were applied after the input video was stabilized.

A straightforward idea for video motion deblurring is to first identify blurry frames, and then apply existing single or multiple image deblurring techniques to them. Unfortunately, we found that existing image deblurring approaches are incapable of generating satisfactory results on video. An example is shown in Fig. 1, where we apply a recent single image deblurring method [Cho and Lee 2009] and a multi-frame deblurring approach [Li et al. 2010] to deblur the blurry frame shown in Fig. 1a. Both results contain significant, unacceptable artifacts due to several reasons. First, the blur kernels in video are both spatially and temporally varying, introduced by both camera and object motions. Recent deblurring methods could effectively handle general camera motions, but object motions would still prevent reliable and accurate blur kernel

estimation for all frames. Second, even with good kernel estimation, deconvolution is sensitive to various outliers, such as noise and saturated pixels, and can easily introduce severe ringing artifacts, as suggested by Cho et al. [2011]. Third, while most previous multi-frame deblurring approaches require input images to be well aligned [Chen et al. 2008; Li et al. 2010], it is usually impossible to accurately align video frames due to moving objects and depth differences. Last but not least, video deblurring demands temporal coherence. Directly applying an image deblurring method to individual frames can easily generate temporally inconsistent results.

In this paper, we present an effective, practical solution for video motion deblurring, which avoids applying direct kernel estimation and deconvolution to video frames. Our method is built upon the key observation that camera shake usually comes from high frequency, irregular hand motion. It causes the same image content to appear sharper on some frames when the motion velocity is small, and more blurry on others when the velocity is large. With proper alignment and motion compensation, sharp regions can be directly used to restore their corresponding regions in blurry frames.

In our approach, we first estimate a parametric, homography-based motion for each frame as an approximation to the real motion, which can be far more complicated due to parallax. We then use the approximated motion for defining the *luckiness* of a pixel to measure its sharpness. To deblur a video frame, we search for luckier pixels in nearby frames and use them to replace less lucky ones in the current frame. The pixel correspondence across frames is obtained using the estimated homographies, followed by a local search of the best matching image patch to compensate for the inaccuracy of the motion model. To compare a sharp patch with a blurry one, we use forward convolution to blur the sharp patch with the estimated blur function of the blurry patch. When copying lucky pixels to a blurry frame, we adopt a patch-based texture synthesis approach [Kwatra et al. 2005] to better preserve object structures. Finally, we impose similarity constraint on the corresponding patches in consecutive frames to maintain temporal coherence of the deblurred frames.

Our method is designed to remove only the blur introduced by camera motion on static objects in the video. This is in lieu with the design goal of video stabilization approaches which usually aim at stabilizing the background only. For moving objects, since their blur is typically dominated by object motion which is preserved in the final video, the blur on them rarely stands out as a noticeable artifact. On the contrary, we found that human perception is much more sensitive to the blur on background regions which are supposed to be still or slowly changing. Our method can effectively remove the most annoying background blur and can work reliably well when moving foreground objects present, as we will demonstrate in Sec. 5.

2 Related Work

Single image deblurring Deblurring a single input image has been extensively studied in recent years. Most success in this area has come with uniform deblurring approaches [Fergus et al. 2006; Shan et al. 2008; Cho and Lee 2009; Levin et al. 2011]. These methods assume a single uniform kernel for the whole image, thus cannot be directly applied to video frames where the blur is spatially varying. Recently, the research focus has been shifted to exploring non-uniform deblurring methods. Whyte et al. [2010] and Hirsch et al. [2011] directly used a 3D blur kernel to represent spatially-varying 2D blur kernels across the image, while Gupta et al. [2010] proposed to represent camera motion using a motion density function. However, in practice we found that these approaches are not robust enough to handle real-world videos we address in this paper, due to many factors such as moving objects, image noise and compression artifacts.

Multi-image deblurring Several approaches have been proposed to jointly deblur multiple blurry images of roughly the same scene. Cai et al. [2009] proposed a numerically stable multi-image deblurring method which explicitly models image registration errors. However this method assumes a uniform blur kernel for each image. Agrawal et al. [2009] varied the exposure time for video frames to make multi-frame deblurring invertible, but it requires special capturing hardware and cannot be applied to normal video sequences. Cho et al. [2007] segmented images into regions of homogeneous motions, and estimated the corresponding motion PSFs to restore latent images, all in a joint energy minimization framework. However, this method handles only 1D Gaussian kernels and lacks the ability to model general camera motion which is common in a video sequence. Li et al. [2010] proposed a system to create a sharp panorama from a motion-blurred video. Their system uses homographies as the motion model between adjacent frames, which leads to spatially-varying blur kernels. The motion and duty cycle parameters are estimated along with latent images in an energy minimization formulation. Our method adopts a similar homography-based motion model for the approximate blur model. However we do not treat the homography-based model as an accurate one, and explicitly handle the model inaccuracy by incorporating local search of matching patches. Furthermore, our system do not use deconvolution to restore latent frames, thus avoids introducing ringing artifacts that are common in previous approaches.

Video deblurring by interpolation Matsushita et al. [2006] proposed a practical video deblurring method in their video stabilization system. They detected sharp frames using the statistics of image gradients, and interpolated sharp frames to increase the sharpness of blurry frames. This is the closest work to our method. However, their frame-to-frame pixel alignment method uses only the camera motion represented by homographies, and does not consider either the effect of blur kernels or the alignment errors introduced by using homographies only. This alignment inaccuracy in their method inevitably degrades the quality of deblurred frames, as we will show in Sec. 5.

Lucky imaging Lucky imaging, also called lucky exposures, is a well-known technique in astronomical photography dated back in 70s [Fried 1978], where a few best images out of many are chosen and combined into a single image to avoid atmospheric turbulence. The similar concept has been recently applied for obtaining a sharp image of a distant object from multiple shots [Joshi and Cohen 2010]. They assumed the camera is static, thus the small amount of misalignment between images can be removed by a patch based search with simple comparison of pixel values. This is however not the case for video deblurring. As the camera motion is intense at times, aligning video frames becomes nontrivial due to different blur amounts of pixels, which prohibit the direct use of pixel values in a patch based search. Furthermore, previous lucky imaging techniques throw away most data and produce only one best image, while for video deblurring we have to restore all frames.

Patch-based synthesis Patch-based sampling methods have achieved state-of-the-art results in a wide range of applications such as texture synthesis [Efros and Freeman 2001], denoising [Buades and Coll 2005; Liu and Freeman 2010], super-resolution [Freedman and Fattal 2011], and interactive image editing [Barnes et al. 2009]. For searching the nearest neighbor given an image patch, previous methods mostly use the sum of squared differences (SSD) as the patch distance metric, and only allow searching in the translation space. Recently Barnes et al. [2010] extended their PatchMatch algorithm to include searching across scales and rotations as

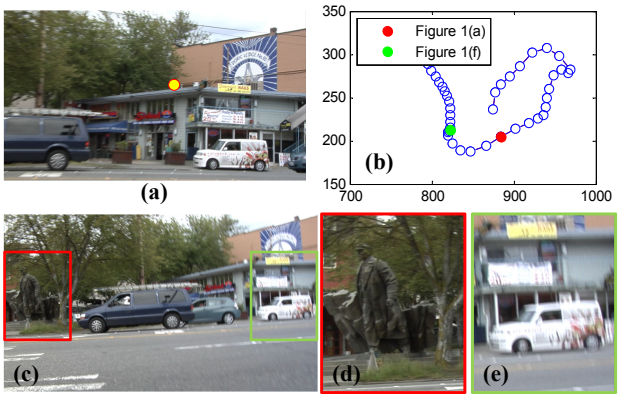

Figure 2: *Sources of sharp regions in a motion-blurred video. (a) A feature point to be tracked for the "cars" video. (b) The trajectory of the feature point in the entire video. (c) One frame of the "cars" video with camera rotation. (d) A local region near the rotation center. (e) A local region far from the rotation center.*

well. However this extension is still insufficient to search across large motion blurs, which is essential to our deblurring framework. HaCohen et al. [2011] proposed a non-rigid dense correspondence algorithm and applied it to image deblurring, by iteratively performing blind-deconvolution and image registration. However this method assumes a global uniform kernel and uses deconvolution to produce the final result, thus shares the same limitations with previous single image deblurring approaches.

Our approach is significantly different from and more robust than previous image deblurring approaches, as it does not rely on accurate kernel estimation and deconvolution. Our method estimates kernels but uses them only for simulating blurring of sharp pixels in a patch-based search, not for deconvolution to obtain the final deblurred frames. Consequently, the required accuracy of the estimated kernels is less than in previous approaches. Our approach is also significantly different from previous work of video deblurring by interpolation, lucky imaging, and patch-based synthesis, with a unique contribution. We convolve a sharp patch with a blur kernel to accurately compare it with a blurry patch. This is a key step, as directly comparing a sharp patch to a blurry patch using pixel differences will fail in finding the most adequate sharp patch.

3 Motion-blurred Video Frames

In this section, we first analyze why our assumption on the existence of sharp image regions in a motion-blurred video is valid. We then describe an approximate blur model which we will use in our deblurring approach.

3.1 Sources of sharp regions

Our method targets on removing motion blurs introduced by hand-held cameras. Hand shake is an irregular motion which changes velocity in a short period of time. The captured video frames sample the hand shake motion at a much higher frame rate. If the video lasts for a reasonable amount of time (typically a few seconds), there are bound to be *lucky* frames captured when the camera motion is almost steady (e.g., turning points of camera motion), resulting in sharp frames, as well as blurry ones which were captured with large camera motion. Fig. 2 shows an example. We selected one representative feature point on the background, shown as the yellow dot in Fig. 2a, and plotted its trajectory in the image space in Fig. 2b. Since the background is static, this trajectory represents the camera

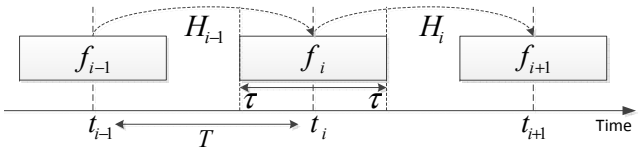

Figure 3: *Illustration of our approximate blur model.*

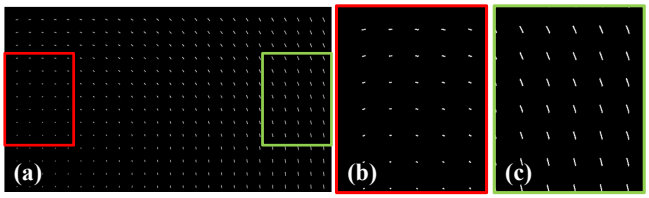

Figure 4: *Estimated blur functions using our approximate blur model for the video frame in Fig. 2c.*

motion very well. It clearly shows that the velocity of the camera motion changes dramatically in the course of the video, producing both sharp (Fig. 1f) and blurry (Fig. 1a) frames at different times.

In addition to sharp and blurry frames, different image regions in a single frame can be sharp and blurry, due to spatial-varying blur kernels. Fig. 2c shows a video frame captured with camera rotation, where the rotation center is around the middle of the left image boundary. An image region near the rotation center is sharp (Fig. 2d) while a region far away from the rotation center is more blurry (Fig. 2e). This example suggests that we should consider local sharpness in video frames for deblurring.

We would like to point out that this assumption fails if the camera motion is dominated by a carefully directed large motion throughout the entire video, such as constant panning. In practice we found that most amateur videos taken with hand-held cameras usually contain shaky camera motions such as walking, thus our assumption holds well on a wide range of videos.

3.2 Approximate blur model

Our method approximates the blurs of video frames using homographies. As shown in Fig. 3, suppose the duty cycle of the camera is 2τ, and the exposure time interval for frame f_i is $[t_i - \tau, t_i + \tau]$. Let the time interval be $T = t_i - t_{i-1}$, and the latent image of f_i be l_i. Assume that the motion between adjacent frames can be approximated by a homography, i.e., $l_{i+1} = \hat{H}_i(l_i)$, where $\hat{H}_i$ is an warping function parameterized by a 3×3 homography matrix H_i, and that the velocity of the motion is constant between adjacent frames. Then, we have

$$f_i = \frac{1}{2\tau}\int_{t=0}^{\tau}\left(\hat{H}_{i-1}^t(l_i) + \hat{H}_i^t(l_i)\right)dt, \qquad (1)$$

where $\hat{H}_{i-1}^t$ and $\hat{H}_i^t$ are warping functions parameterized by the homography matrices H_{i-1}^t and H_i^t, respectively, which are defined as:

$$H_{i-1}^t = \frac{T-t}{T}I + \frac{t}{T}H_{i-1}^{-1}, \; H_i^t = \frac{T-t}{T}I + \frac{t}{T}H_i. \qquad (2)$$

I is the 3×3 identity matrix and H_{i-1}^{-1} is the inverse matrix of H_{i-1}. Discretizing Eq. (1) gives us:

$$f_i \doteq b_i(l_i) = \frac{1}{1+2\tau}\left[\sum_{d=1}^{\tau}\left(\hat{H}_{i-1}^d(l_i) + \hat{H}_i^d(l_i)\right) + l_i\right], \qquad (3)$$

where T becomes the sampling rate in $[t_i, t_{i+1}]$ which we fix as 20 in our implementation, and τ becomes the number of samples that fall into the duty cycle. We call b_i the *blur function* for frame i. Fig. 4 shows the estimated blur functions for the video frame in Fig. 2c.

This blur model is similar to the one developed in the multi-image deblurring method for panorama generation [Li et al. 2010], which also uses homography as the underlying motion model. However, our work only treats this model as an approximation in order to deal with more complicated videos than panoramas, while it was treated as an accurate model in the previous method. To explicitly handle the modeling errors, our method employs an additional local search step for aligning image regions of different frames, as we will describe in detail in Sec. 4.1.

3.3 Luckiness measurement

With the motion model in Sec. 3.2, we introduce a measurement of *luckiness* for a pixel in a video frame, which describes the absolute displacement of the pixel among adjacent frames. For a pixel x in frame f_i, its luckiness is defined as:

$$\alpha_i(x) \quad = \quad \exp\left(-\frac{\|\widetilde{H}_{i-1}^{-1}(x) - x\|^2 + \|\widetilde{H}_i(x) - x\|^2}{2\sigma_l^2}\right), \quad (4)$$

where $\widetilde{H}$ is a function that maps a pixel position to another pixel position according to the homography H. σ_l is a constant which we set as 20 pixels in our implementation. Eq. (4) computes the displacement of pixel x when the camera moves from frame f_{i-1} to f_{i+1} through f_i. When the frame-to-frame motion of x is small, H_{i-1}^{-1} and H_i are close to I, thus $\alpha_i(x)$ is close to 1, indicating that the image patch centered at x is likely to be sharp. Otherwise $\alpha_i(x)$ is small, indicating that the patch is likely to contain large motion blur. The luckiness α_i of a whole frame f_i is simply defined as the average value of all $\alpha_i(x)$ for pixels in f_i.

3.4 Blur function estimation

There are two parameters that we have to estimate for the blur function b_i in Eq. (3) before we use it for deblurring: the homography H_i and the duty cycle τ. To estimate the homography H_i, we first apply feature tracking using a standard KLT approach [Shi and Tomasi 1994], and use the tracked feature points to compute the initial homographies. We then refine the homographies using Lucas-Kanade registration [Baker and Matthews 2004] between frames.

In our deblurring approach, we need to estimate homographies not only between adjacent frames, but also between any two frames in a local temporal window $W_i = [i - M, i + M]$, where M is set to 5 frames in our implementation. The homography from frame i to frame j is denoted by H_{ij}, where $j \in W_i$. Obviously $H_{i,i+1} = H_i$ in Fig. 3. Homographies among non-adjacent frames are initialized and refined in the same way as H_i.

Note that unlike the previous approach [Li et al. 2010], we do not further update the homographes when we deblur the video frames. This is because we only treat the homography as an approximate motion model. Small errors in homography estimation are handled by local search of matching patches in the deblurring process.

To compute τ, we first select a set of frame pairs, where each pair has a large difference in the luckiness measurement, so that the accuracy of blur functions can be effectively tested. Let (f_i^k, f_j^k), $k = 1, ..., K$, be K pairs of frames with $j \in W_i$, where the frame luckiness difference, $\alpha_i - \alpha_j$, is larger than a threshold (0.6α in our system, where $\alpha = \max_i \alpha_i$). We then seek the optimal τ which minimizes:

$$E(\tau) \quad = \quad \sum_{k=1}^{K} \left\| b_j^k \left(\hat{H}_{ij}^k(f_i^k) \right) - f_j^k \right\|^2. \quad (5)$$

Intuitively, we first align f_i^k with f_j^k using the homography H_{ij}^k, then blur the aligned sharp frame $\hat{H}_{ij}^k(f_i^k)$ using the blur function b_j^k of f_j^k, and compute the sum of squared differences between the synthetically blurred frame and the observed f_j^k. The value of Eq. (5) depends on the blur functions b_j^k for all k: if functions b_j^k are accurate, the value should be small, and vise versa. Since b_j^k is determined by Eq. (3) for a given value of τ, Eq. (5) becomes an energy function of τ. We minimize Eq. (5) using a brute-force search as τ can take only a limited set of integer values from 1 to $\lfloor T/2 \rfloor$.

4 Video Frame Deblurring

4.1 Single frame deblurring

Once we have obtained the blur function b_i for frame f_i, we could compute the latent frame l_i using a deconvolution method that can handle non-uniform blurs represented by homographies [Li et al. 2010; Tai et al. 2011]. However, as discussed in Sec. 1, this straightforward approach generates less satisfactory results in practice. We instead use lucky patches in nearby frames to restore l_i, avoiding artifacts from deconvolution.

Patch deblurring Let $f_{i,x}$ be an $n \times n$ image patch in a frame f_i centered at a pixel x. In our implementation, $n = 11$. We deblur $f_{i,x}$ by computing an weighted average of sharp patches from nearby frames f_j in the temporal window W_i. That is,

$$l_{i,x} \quad = \quad \frac{1}{Z} \sum_{(j,y) \in \Omega_{i,x}} w(i,x,j,y) f_{j,y}, \quad (6)$$

where $l_{i,x}$ is a deblurred patch of $f_{i,x}$, and $f_{j,y}$ is a patch in the *warped* frame $\hat{H}_{ji}(f_j)$ centered at a pixel y. The weight $w(i,x,j,y)$ is defined as:

$$w(i,x,j,y) \quad = \quad \exp\left(-\frac{\|b_{j,y} - f_{i,x}\|^2}{2\sigma_w^2}\right), \quad (7)$$

where $b_{j,y}$ is a patch centered at y in the *blurred* warped frame $b_i\left(\hat{H}_{ji}(f_j)\right)$, which is obtained by applying the blur function b_i of the current frame f_i to $\hat{H}_{ji}(f_j)$, and σ_w is a constant set to 0.1 in our system. $\Omega_{i,x}$ is a set of matching patches for $f_{i,x}$ sampled from warped nearby frames $\hat{H}_{ji}(f_j)$. Z is the normalization factor, i.e., $Z = \sum_{(j,y) \in \Omega_{i,x}} w(i,x,j,y)$.

The weight $w(i,x,j,y)$ is analogous to the data fitting term of previous deblurring methods, which measures the difference of the latent image from the input blurry image when it has been blurred using the estimated kernel. In our approach, we test a warped patch $f_{j,y}$ from a nearby frame f_j for its eligibility to be the latent patch $l_{i,x}$, by comparing it with the input blurry patch $f_{i,x}$ after blurring it with the estimated blur function b_i. The weight $w(i,x,j,y)$ becomes high as the blurred patch $b_{j,y}$ matches with $f_{i,x}$. Consequently, warped patches $f_{j,y}$ have more contributions in the weighted averaging in Eq. (6) when they are similar to the latent patch $l_{i,x}$.

To determine the patch set $\Omega_{i,x}$, we find the N best-matching patches from warped nearby frames $\hat{H}_{ji}(f_j)$, where $j \in W_i$. We

Figure 5: *Illustration of the impact of each step in our algorithm on the final result. (a) A region in the input blurry frame. (b) Copying center pixels of matched patches from nearby frames. (c) Adding weighted patch averaging in Eq. (9). (d) Adding the lucky patch prior in Eq. (10). (e) Adding the frame processing order.*

use $w(i,x,j,y)$ as the degree of matching, which is equivalent to selecting patches by solving

$$\underset{j,y}{\operatorname{argmin}} \|b_{j,y} - f_{i,x}\|^2. \tag{8}$$

For the search range of y to find a matching patch $f_{j,y}$ in $\hat{H}_{ji}(f_j)$, we use an $m \times m$ window centered at the pixel x. Ideally, if H_{ji} accurately estimates the motion from f_j to f_i, we can simply set the search range m to be one, i.e., only using the patch centered at x in $\hat{H}_{ji}(f_j)$. However, in practice due to parallax and object motions, the real motion among frames is generally more complicated than a single homography. Using a larger value than one for the search range m allows us to compensate for the error in our motion model using homographies. In our implementation, we use $m = 21$, which seems large enough to find a good patch, given that a homography could still be a good estimate for the motion between frames.

In our experiments, we found that using a large N often leads to over-smoothed deblurring results, due to the small misalignments among the N best-matching patches. We thus use $N = 1$ for best sharpness, and then Eq. (6) is reduced to simply using the best-matching patch to restore a latent patch. As mentioned in Sec. 3.4, we set $M = 5$ for the temporal window W_i, meaning that ten nearby frames and f_i itself are included in the patch search.

Frame deblurring To restore the latent frame l_i from a blurry input frame f_i, we could simply perform the patch deblurring using Eq. (6) at each pixel x in f_i and keep the center pixels of the deblurred patches. However, this approach may incur misalignments of object structures in l_i, as pixels in l_i are determined individually without enforcing spatial coherence (see Fig. 5b). Instead, we adapt a patch-based texture synthesis approach [Kwatra et al. 2005] to merge the effects of overlapping deblurred patches in l_i.

Let $l_i(x)$ be the value of l_i at a pixel x. We determine $l_i(x)$ as:

$$\begin{aligned} l_i(x) &= \frac{1}{Z} \sum_{x' \in \Omega_x} Z_{x'} l_{i,x'}(x), \\ &= \frac{1}{Z} \sum_{x' \in \Omega_x} \sum_{(j,y) \in \Omega_{i,x'}} w(i,x',j,y) f_{j,y}(x), \end{aligned} \tag{9}$$

where Ω_x is the set of pixels x' in the $n \times n$ spatial window centered at x for which deblurred patches $l_{i,x'}$ have been derived using Eq. (6), and $l_{i,x'}(x)$ is the pixel value of $l_{i,x'}$ at x. Z is the normalization factor, i.e., $Z = \sum_{x' \in \Omega_x} Z_{x'}$, where $Z_{x'} = \sum_{(j,y) \in \Omega_{i,x'}} w(i,x',j,y)$. $f_{j,y}(x)$ is the value of pixel x in a warped patch $f_{j,y}$. If we compute a deblurred patch for every pixel in f_i, there will be n^2 pixels in Ω_x for a pixel x in f_i except around the image boundary. Then, pixel x will be covered by n^2 deblurred patches whose values at x are weighted averaged using Eq. (9) to determine $l_i(x)$. To accelerate the frame deblurring process, we perform patch deblurring only for a sparse regular grid of pixels, not for every pixel, as done in [Kwatra et al. 2005]. This sparse sampling also helps avoiding over-smoothed deblurring results, which can be caused by averaging many patches. Fig. 5c shows the deblurred result with better preserved object structures.

Handling moving objects Our deblurring method can successfully process slightly moving objects, due to the local search of matching patches. In contrast, our method keeps objects with large motions almost untouched in the deblurring process. When a patch $f_{i,x}$ belongs to a moving object in f_i, the object motion incurs a different blur from the blur function b_i, and would dominate the true blur function for $f_{i,x}$ if the motion is large. In this case, due to the blur function difference, the local patch search cannot find a matching patch in another frame f_j with a small fitting error defined in Eq. (8). On the other hand, the fitting error between $f_{i,x}$ and $b_i(f_{i,x})$ is relatively small, since $f_{i,x}$ is already severely blurred by the object motion, and $b_i(f_{i,x})$ is a slightly smoothed version of $f_{i,x}$ with the same appearance. Thus $f_{i,x}$ becomes the best matching patch for itself, and the latent patch $l_{i,x}$ computed by Eq. (6) would remain similar to $f_{i,x}$.

4.2 Improved deblurring using luckiness

To further improve the sharpness of deblurred frames, we introduce a new weight $w'(i,x,j,y)$ with which sharper patches are preferred over blurrier ones:

$$w'(i,x,j,y) = w(i,x,j,y) \cdot \exp\left(-\frac{\|\mathbf{1} - \alpha_{j,y}\|^2}{2\gamma^2}\right), \tag{10}$$

where $\alpha_{j,y}$ is an $n \times n$ patch centered at a pixel y in a luckiness map α_j, and γ is a constant. A luckiness map α_j consists of the luckiness values of pixels in the warped frame $\hat{H}_{ji}(f_j)$. $\mathbf{1}$ is an $n \times n$ patch whose elements are all one. As we use the new weight $w'(i,x,j,y)$ for selecting the best-matching patches, Eq. (8) becomes:

$$\underset{j,y}{\operatorname{argmin}} \left\{ \|b_{j,y} - f_{i,x}\|^2 + \lambda \|\mathbf{1} - \alpha_{j,y}\|^2 \right\}, \tag{11}$$

where $\lambda = \sigma^2/\gamma^2$. We use a small value for λ, which is 0.01 in our implementation. This allows the patch match term to dominate Eq. (11) and the lucky patch prior to have effect only when the values of the patch match term are similar among different patches. Fig. 5d shows that using luckiness helps improve the sharpness of a restored frame.

We also use the luckiness values to determine the processing order of frames when applying our frame deblurring method to the whole video sequence. We first sort all frames based on their luckiness values, and start processing from the luckiest frame first. For luckier frames, most pixels will remain unchanged after deblurring. As the luckiness values of frames become lower, more pixels will be

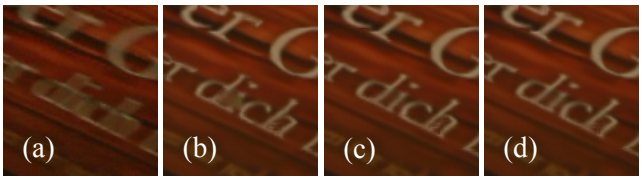

Figure 6: *Results at iterations of the iterative improvement. (a) An input frame. (b) 1st iteration. (c) 2nd iteration. (d) 3rd iteration.*

updated by sharper pixels from already processed frames. We treat the luckiness as another color channel and update it in the same way as we update RGB values of pixels. As a result, sharp pixels are propagated from luckier frames to less lucky ones. Fig. 5e shows the final deblurring result using the proposed frame processing order. It is clearly sharper than Fig. 5d which was generated without using the processing results of other frames.

4.3 Iterative improvement of deblurred frames

In patch-based texture synthesis [Kwatra et al. 2005] and image retargeting [Simakov et al. 2008], the synthesis is performed iteratively to obtain more coherent textures and object structures. Our method takes a similar iterative process to improve the spatial coherence of deblurred frames. We use the result frames from the previous iteration of patch search and deblurring in the current iteration. As a result, the pixel sharpness and the object structures in the deblurred frames become enhanced with the iterations.

For iterative improvement, we first modify Eq. (6) as:

$$l_{i,x}^{p+1} = \frac{1}{Z} \sum_{(j,y)\in\Omega_{i,x}} w'(i,x,j,y) l_{j,y}^{p}, \tag{12}$$

where l_i^p is the i-th *latent* frame restored in the p-th iteration, and $l_i^0 = f_i$. We also modify Eq. (10) as:

$$w'(i,x,j,y) = \exp\left(-\frac{\|b_{j,y}^p - f_{i,x}\|^2}{2\sigma_w^2}\right) \exp\left(-\frac{\|\mathbf{1} - \alpha_{j,y}^p\|^2}{2\gamma^2}\right), \tag{13}$$

where $b_{j,y}^p$ is a patch in a blurred warped latent frame $b_i\left(\hat{H}_{ji}(l_j^p)\right)$, and $\alpha_{j,y}^p$ is a patch in a luckiness map α_j^p updated in the p-th iteration. Fig. 6 shows an example of the iterative improvement.

The number of iterations needed for iterative improvement is directly related to how far away the sharp frames are distributed from each other. It can be automatically determined based on the temporal window size, $2M+1$, which basically defines how far a sharp frame can expand in one iteration. As shown in Fig. 7, we compute the luckiness values of all frames in the input video, and find all sharp frames whose luckiness values are higher than a high threshold (0.85 in our system). We then find the maximum distance between two adjacent sharp frames, denoted by M_s. The number of iterations can be computed as $\lceil (M_s - 1)/2M \rceil$.

4.4 Improving temporal coherence

Although there is no explicit temporal coherence term involved in Eq. (13), the resulting video generated by the iterative deblurring process in Sec. 4.3 is mostly temporally coherent. This is due to the fact that we constrain the patch search to be in a small spatial window after homography registration, and thus for the same blurry patch in consecutive frames, the same sharp patch tends to be selected in the local search of matching patches. Furthermore,

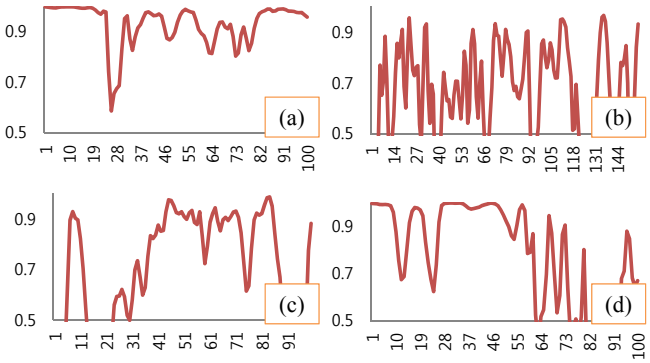

Figure 7: *Frame luckiness values of four different motion-blurred videos. The proportions of sharp frames (i.e., $\alpha_i > 0.85$) are (a) 88% (b) 21% (c) 42%, and (d) 54%.*

the iterative process encourages a sharp patch to propagate across multiple frames, achieving better temporal coherence.

To further improve the temporal coherence, after the iterative improvement of deblurred frames, we have additional iterations with slight modification of Eq. (12). That is, we restrict the temporal window W_i as $W_i = [i-1, i+1]$ and use three patches $l_{i-1,x_{i-1}}^p$, $l_{i,x}^p$, and $l_{i+1,x_{i+1}}^p$ for computing Eq. (12), where x_{i-1} and x_{i+1} are the matched pixel positions of x in the previous and next frames, respectively, which maximize the weighting function in Eq. (7). This is equivalent to finding a temporally smooth video frames l_i' from given video frames l_i by optimizing the following Markov random field based energy function:

$$\begin{aligned} E_t(\{l_i'\}) &= \sum_{i,x} w(i,x,i,x)\|l_{i,x}' - l_{i,x}\|^2 \\ &+ \sum_{i,x} \sum_{j\in\{i-1,i+1\}} w(i,x,j,x_j)\|l_{i,x}' - l_{j,x_j}'\|^2 \end{aligned} \tag{14}$$

with the gradient descent method where the patch correspondence and weights are updated at each gradient descent step. In Eq. (14), the first term on the right hand side is a data fidelity term, which forces l_i' to be similar to the given frame l_i, and the second and third terms encourage the updated frames l_i' to be close to their neighboring frames. Consequently, the resulting frames are close to the given deblurred frames l_i as well as temporally coherent. We typically iterate this step twice to obtain the final result frames.

3D volumetric patches have been used for maintaining temporal coherence in video retargeting [Simakov et al. 2008]. One possible idea for temporal coherence in our method is to extend a 2D patch into neighboring frames to construct a 3D patch, using homographies to align corresponding pixels. However, the requirement for temporal coherence is different among video retargeting and our method. In video retargeting, 3D volumetric patches in the original video are desired to be copied into the result video while preserving the order of consecutive frames. In contrast, in our method, patch search is performed to find sharp patches for restoring blurry patches, and comparison of two 3D volumetric patches is meaningful only when the sharpness and the blur function are consistent in all the consecutive frames involved in the 3D patches. In practice, real videos have dynamically varying sharpness and blur functions among frames, and 3D volumetric patches would not be useful for our local search of matching patches.

Figure 8: *Examples of deblurred video frames. Top: Input blurry frames. Bottom: Our deblurring results. Full video sequences are included in the supplementary material.*

5 Results and Comparisons

As our method is based on the assumption of the existence of sharp frames, we plot frame luckiness values of a few video captured by hand-held cameras to see how often sharp video frames actually appear in real motion-blurred videos (Fig. 7). We empirically found that frames of luckiness values greater than 0.85 are not visually blurry, i.e., sharp frames. While the shape of the plot varies among the test videos, all plots suggest that a large number of sharp frames exist in a motion-blurred video. Furthermore, as some blurry frames may have sharp regions due to non-uniform blur as described in Sec. 3.1, available sharp regions have a denser distribution than sharp frames.

We have applied our method to a variety of example videos shot by hand-held cameras. In the supplementary material, we provide stabilized versions of the input and deblurred videos, which allow us to concentrate on the blur artifact without being distracted by shaky camera motion. As can be seen in the examples, the stabilized input videos contain annoying blur artifact caused by original camera shake, where video frames suddenly become blurry for a few frames before coming back to normal. Our method can largely remove this artifact and generate temporal-coherently sharp videos.

Fig. 8 shows input blurry frames from four different videos and the deblurring results. These examples demonstrate that our method can successfully restore sharp frames from blurry input. Note that these are challenging examples given the moving objects and significant depth differences among the objects. Nevertheless, our method recovers sharp details without noticeable artifacts.

Fig. 9 visualizes pixel luckiness values of one frame at each step of our algorithm. Initially, the luckiness values of all pixels, computed by Eq. (4), are low, indicating the frame is blurry. As iteration goes, the updated luckiness values become high for more pixels. Fig. 10 shows frame luckiness values of the entire video. The input video has dynamically changing frame luckiness values due to irregular camera motion. The frame-wise restoration step improves the frame luckiness for all blurry frames. The final step for improving temporal coherence produces smoothly changing frame luckiness values.

In Fig. 11 we compare our method with the previous deblurring methods. Figs. 11a and 11d show results of a deconvolution-based deblurring approach. Similar to Li et al. [2010], we first estimate homographies between frames, and use them to align every three consecutive frames. We then approximate spatially-varying motion blur kernels using the estimated homographies between frames, and finally apply the recently proposed projective motion Richardson-Lucy algorithm [Tai et al. 2011] to recover the latent

Figure 9: *Visualization of pixel luckiness values at each step of our algorithm. From left to right: input blurry frame; after the 1st iteration; after the 2nd iteration.*

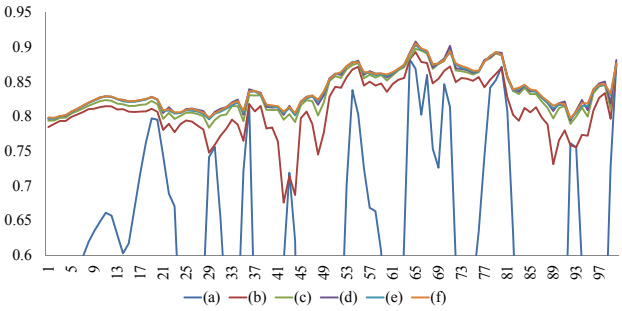

Figure 10: *Plot of frame luckiness values at each step of our algorithm. (a) Input blurry video. (b)-(d) 1st, 2nd, and 3rd iterations, respectively. (e) & (f) 1st and 2nd iterations for improving temporal coherence.*

image. This implementation does not exactly match the method of Li et al. [2010]. Nevertheless, the implementation shares the core limitations with their method, i.e., a limited homography-based motion model and no handling of moving objects.

The input video of Fig. 11 presents strong parallax due to the significant depth difference between the front pole and the background street, thus the adjacent frames cannot be aligned well using a single homography. Consequently, the deblurring result in Fig. 11a shows noticeable artifacts around the front pole. In Fig. 11d, pixels in moving objects cannot be aligned using a single homography either, leading to severe ringing artifacts around the moving car. In contrast, our method generates better results in both cases, as it naturally handles errors from the homography-based motion model and moving objects.

Figs. 11b and 11e show results of the interpolation-based video

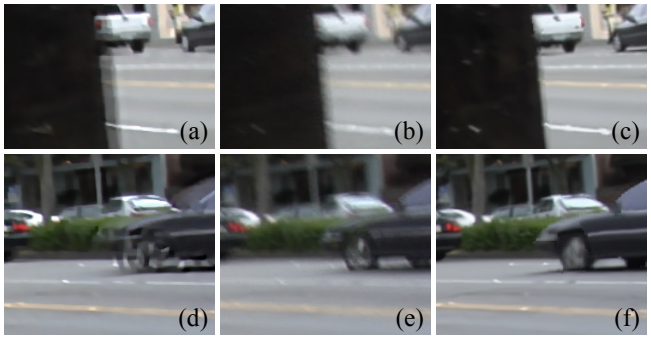

Figure 11: *Comparison with previous deblurring approaches. (a) & (d) Multi-frame deblurring results. (b) & (e) Matsushita et al. [2006]. (c) & (f) Our results.*

Figure 12: *From left to right: input blurry frame, shock filtering result of the input frame, our method, our method followed by shock filtering, and Matsushita et al. [2006] followed by shock filtering.*

deblurring method of Matsushita et al. [2006]. As their method uses simple frame-interpolation without considering the characteristics of underlying motion blur, the result still appears to be blurry. Their method also demands accurate frame alignment for high quality results, which is hardly achievable in practice. In contrast, our method generates better results by involving the estimated blur function and compensating for frame alignment error in Eq. (13).

Due to latent patch averaging in Eq. (12), our deblurring result may look a bit too smooth. However, as large motion blurs in video frames have been removed by our method, the perceived sharpness of our result can easily be enhanced by simple image filtering. We did not apply any filtering to our deblurring results shown in this paper, except Fig. 12 that demonstrates the effect of shock filter [Osher and Rudin 1990] applied to our deblurring result. Note that the input blurry frame and the result of Matsushita et al. [2006] are not improved at all when the same filter is applied to them. For the deblurring results included in the supplementary video, we applied shock filter for enhanced visual quality.

Improving video stabilization Our method can improve stabilization quality of a video, as feature points can be located more accurately in the deblurred frames. Fig. 13 shows consecutive frames of stabilized videos. The top row is the stabilization result using the original frames, and the bottom row is the result using the deblurred frames, all generated by the same algorithm [Liu et al. 2011] with the same set of settings. As the original video frames are blurry, a feature-point-based stabilization method may fail to match feature points between frames. As a result, the top row shows a sudden jump between the two consecutive stabilized frames. In contrast, the bottom row shows a smooth transition between the same two frames. We refer the reader to the supplementary material for comparison of the stabilization results.

5.1 Failure cases

Our approach has a few limitations. First, feature tracking may fail when there are large moving objects or significant depth variation. This may cause failure in estimating blur function, and in finding

Figure 13: *Stabilization results of a blurry video (top row) and the deblurred video (bottom row). The left two columns show two consecutive frames of stabilized results. The right column shows difference between the left two frames. While the result of the blurry video shows a sudden jump due to incorrect feature point matching caused by motion blur in the frames, the stabilization result of the deblurred video shows smooth transition between the frames.*

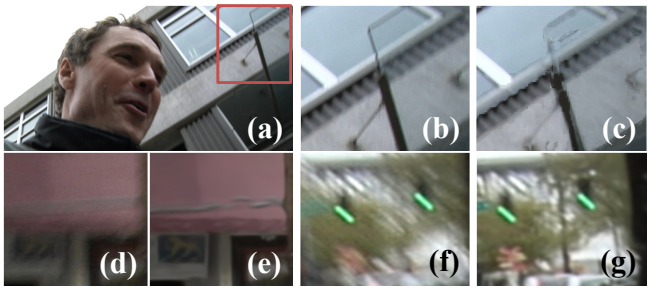

Figure 14: *Examples of failure cases. (a) Input video frame. (b) Magnified view of an input region. (c) Our result. (d) & (f) Input frames. (e) & (g) Our results.*

proper sharp patches in nearby frames (Figs. 14a-c). In addition, for a severely blurred patch, the fine image structure in it might have been destroyed completely, resulting in incorrect patch finding. Figs. 14d-e show such an example. Although the input frame is largely improved by our method, we can still see noticeable artifact around a fine image structure.

Currently our method has difficulty to handle saturated pixels. Due to clipping of pixel values, the motion blur model in Eq. (1) does not hold for saturated pixels, and our method cannot properly match a sharp patch with a blurry one in the presence of saturated pixels. Fig. 14f shows such an example, where the traffic lights cause pixels to be saturated around them. Our method fails to improve the regions around the traffic lights, as shown in Fig. 14g, although other parts of the frame is largely improved.

Our method is an example-based approach which relies on using sharp patches from nearby frames to restore blurry ones. If the camera motion is constantly large and there is no sharp patches available, our method cannot restore blurry frames, just leaving the input video untouched. A similar situation is that only parts of the scene have sharp patches available, such as a video where the camera pans at the beginning, then stops at the end. Combining our method with a deconvolution-based approach using multi-frames may help handle these limitation cases.

6 Discussion and Future Work

The camera motion in a hand-held sequence often causes some portion of video frames to be more blurry than others. We present a practical video deblurring method to restore sharp frames. Since our solution only involves forward convolution and patch-based im-

age synthesis, it is robust enough to handle a wide range of real-world videos. This is in sharp contrast to previous deconvolution-based deblurring methods, which are incapable of dealing with various common outliers, such as moving objects, noise, and compression artifacts.

As pointed out in Sec. 5.1, our method still has a few limitations that we would like to resolve in future. Particularly, we plan to develop a more complicated patch matching algorithm which can handle saturated pixels, by explicitly modeling saturated pixels as a non-linear operation in Eq. (1). We also plan to incorporate deconvolution techniques into our method, so that we can improve moving objects by separating the camera motion from the object motion, and remove the effect of the former.

Currently our method is implemented in C++ and runs on a single thread. Deblurring a HD size video frame on a PC with Intel Core i7 CPU takes about one minute. However, our approach is easily parallelizable, as the search of sharp patches for blurry pixels can be carried out independently. As future work we plan to implement our method on GPU to dramatically improve the performance.

Acknowledgements We thank the anonymous reviewers for their constructive comments, and David Simons for initial discussion. This work was supported in part by Industrial Strategic Technology Development Program of KEIT (KI001820) and Brain Korea 21 Project.

References

AGRAWAL, A., XU, Y., AND RASKAR, R. 2009. Invertible motion blur in video. *ACM Trans. Graphics 28*, 3, 95:1–95:8.

BAKER, S., AND MATTHEWS, I. 2004. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision (IJCV) 56*, 3, 221–255.

BARNES, C., SHECHTMAN, E., FINKELSTEIN, A., AND GOLDMAN, D. B. 2009. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graphics 28*, 3, 24:1–24:11.

BARNES, C., SHECHTMAN, E., GOLDMAN, D. B., AND FINKELSTEIN, A. 2010. The generalized PatchMatch correspondence algorithm. In *Proc. ECCV 2010*, 29–43.

BUADES, A., AND COLL, B. 2005. A non-local algorithm for image denoising. In *Proc. CVPR 2006*, 60–65.

CAI, J.-F., JI, H., LIU, C., AND SHEN, Z. 2009. Blind motion deblurring using multiple images. *J. Comput. Phys. 228*, 5057–5071.

CHEN, J., YUAN, L., TANG, C.-K., AND QUAN, L. 2008. Robust dual motion deblurring. In *Proc. CVPR 2008*, 1–8.

CHO, S., AND LEE, S. 2009. Fast motion deblurring. *ACM Trans. Graphics 28*, 5, 145:1–145:8.

CHO, S., MATSUSHITA, Y., AND LEE, S. 2007. Removing non-uniform motion blur from images. In *Proc. ICCV 2007*, 1–8.

CHO, S., WANG, J., AND LEE, S. 2011. Handling outliers in non-blind image deconvolution. In *Proc. ICCV 2011*, 495–502.

EFROS, A. A., AND FREEMAN, W. T. 2001. Image quilting for texture synthesis and transfer. *Proc. ACM SIGGRAPH 2001*, 341–346.

FERGUS, R., SINGH, B., HERTZMANN, A., ROWEIS, S. T., AND FREEMAN, W. T. 2006. Removing camera shake from a single photograph. *ACM Trans. Graphics 25*, 3, 787–794.

FREEDMAN, G., AND FATTAL, R. 2011. Image and video upscaling from local self-examples. *ACM Trans. Graphics 30*, 2, 12:1–12:11.

FRIED, D. L. 1978. Probability of getting a lucky short-exposure image through turbulence. *J. Opt. Soc. Am. 68*, 12, 1651–1657.

GRUNDMANN, M., KWATRA, V., AND ESSA, I. 2011. Auto-directed video stabilization with robust L1 optimal camera paths. In *Proc. CVPR 2011*, 225–232.

GUPTA, A., JOSHI, N., ZITNICK, C. L., COHEN, M., AND CURLESS, B. 2010. Single image deblurring using motion density functions. In *Proc. ECCV 2010*, 171–184.

HACOHEN, Y., SHECHTMAN, E., GOLDMAN, D. B., AND LISCHINSKI, D. 2011. Non-rigid dense correspondence with applications for image enhancement. *ACM Trans. Graphics 30*, 4, 70:1–70:10.

HIRSCH, M., SCHULER, C. J., HARMELING, S., AND SCHÖLKOPF, B. 2011. Fast removal of non-uniform camera shake. In *Proc. ICCV 2011*, 463–470.

JOSHI, N., AND COHEN, M. 2010. Seeing mt. rainier: Lucky imaging for multi-image denoising, sharpening, and haze removal. In *Proc. ICCP 2010*, 1–8.

KWATRA, V., ESSA, I., BOBICK, A., AND KWATRA, N. 2005. Texture optimization for example-based synthesis. *ACM Trans. Graphics 24*, 3, 795–802.

LEVIN, A., WEISS, Y., DURAND, F., AND FREEMAN, W. T. 2011. Efficient marginal likelihood optimization in blind deconvolution. In *Proc. CVPR 2011*, 2657–2664.

LI, Y., KANG, S. B., JOSHI, N., SEITZ, S. M., AND HUTTENLOCHER, D. P. 2010. Generating sharp panoramas from motion-blurred videos. In *Proc. CVPR 2010*, 2424–2431.

LIU, C., AND FREEMAN, W. T. 2010. A high-quality video denoising algorithm based on reliable motion estimation. In *Proc. ECCV 2010*, 706–719.

LIU, F., GLEICHER, M., WANG, J., JIN, H., AND AGARWALA, A. 2011. Subspace video stabilization. *ACM Trans. Graphics 30*, 1, 4:1–4:10.

MATSUSHITA, Y., OFEK, E., GE, W., TANG, X., AND SHUM, H.-Y. 2006. Full-frame video stabilization with motion inpainting. *IEEE Trans. Pattern Analysis Machine Intelligence 28*, 7, 1150–1163.

OSHER, S., AND RUDIN, L. I. 1990. Feature-oriented image enhancement using shock filters. *SIAM Journal on Numerical Analysis 27*, 4, 919–940.

SHAN, Q., JIA, J., AND AGARWALA, A. 2008. High-quality motion deblurring from a single image. *ACM Trans. Graphics 27*, 3, 73:1–73:10.

SHI, J., AND TOMASI, C. 1994. Good features to track. In *Proc. CVPR 1994*, 593–600.

SIMAKOV, D., CASPI, Y., SHECHTMAN, E., AND IRANI, M. 2008. Summarizing visual data using bidirectional similarity. In *Proc. CVPR 2008*, 1–8.

TAI, Y.-W., TAN, P., AND BROWN, M. S. 2011. Richardson-lucy deblurring for scenes under a projective motion path. *IEEE Trans. Pattern Analysis Machine Intelligence 33*, 8, 1603–1618.

WHYTE, O., SIVIC, J., ZISSERMAN, A., AND PONCE, J. 2010. Non-uniform deblurring for shaken images. In *Proc. CVPR 2010*, 491–498.