View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by DSpace at University of West Bohemia

Multiframe Visual-Inertial Blur Estimation and Removal for Unmodified Smartphones

Gábor Sörös, Severin Münger, Carlo Beltrame, Luc Humair
Department of Computer Science
ETH Zurich
soeroesg@ethz.ch, {muengers|becarlo|humairl}@student.ethz.ch

ABSTRACT

Pictures and videos taken with smartphone cameras often suffer from motion blur due to handshake during the exposure time. Recovering a sharp frame from a blurry one is an ill-posed problem but in smartphone applications additional cues can aid the solution. We propose a blur removal algorithm that exploits information from subsequent camera frames and the built-in inertial sensors of an unmodified smartphone. We extend the fast non-blind uniform blur removal algorithm of Krishnan and Fergus to non-uniform blur and to multiple input frames. We estimate piecewise uniform blur kernels from the gyroscope measurements of the smartphone and we adaptively steer our multiframe deconvolution framework towards the sharpest input patches. We show in qualitative experiments that our algorithm can remove synthetic and real blur from individual frames of a degraded image sequence within a few seconds.

Keywords

multiframe blur removal, deblurring, smartphone, camera, gyroscope, motion blur, image restoration

1 INTRODUCTION

Casually taking photographs or videos with smartphones has become both easy and widespread. There are, however, two important effects that degrade the quality of smartphone images. First, handshake during the exposure is almost unavoidable with lightweight cameras and often results in motion-blurred images. Second, the rolling shutter in CMOS image sensors introduces a small time delay in capturing different rows of the image that causes image distortions. Retaking the pictures is often not possible, hence there is need for post-shoot solutions that can recover a sharp image of the scene from the degraded one(s). In this paper, we address the problem of blur removal and rolling shutter rectification for *unmodified* smartphones, i.e., without external hardware and without access to low-level camera controls.

In the recent years, a large number of algorithms have been proposed for restoring blurred images. As blur (in the simplest case) is modeled by a convolution of a sharp image with a blur kernel, blur removal is also termed deconvolution in the literature. We distinguish between non-blind deconvolution, where the blur kernel is known in advance, and blind deconvolution, where the blur kernel is unknown and needs to be estimated first. The blur kernel can be estimated for instance from salient edges in the image [Jos08, Cho09, Sun13, Xu13], from the frequency domain [Gol12], from an auxiliary camera [Tai10], or from motion sensors [Jos10]. Kernel estimation from the image content alone often involves iterative optimization schemes that are computationally too complex to perform on a smartphone within acceptable time. Auxiliary hardware might be expensive and difficult to mount, so kernel estimation from built-in motion sensors seems the most appealing for smartphone applications.

Unfortunately, even known blur is difficult to invert because deconvolution is mathematically ill-posed which means many false images can also satisfy the equations. Deconvolution algorithms usually constrain the solution space to images that follow certain properties of natural images [Kri09]. Another common assumption is uniform blur over the image which simplifies the mathematical models and allows for faster restoration algorithms. However, this is usually violated in real scenarios which can lead the restoration to fail, often even lowering the quality of the processed image [Lev09]. Handling different blur at each pixel of the image is computationally demanding, so for smartphone applications a semi-non-uniform approach might be the best that divides the image to smaller overlapping regions, where uniform blur can be assumed, and restores those regions independently.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

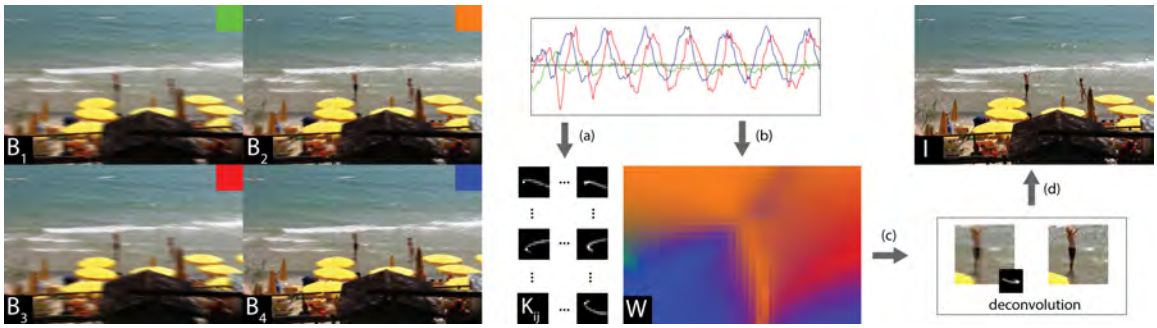

Figure 1: Illustration of our gyroscope-aided multiframe blur removal algorithm. *(a)* First, piecewise uniform blur kernels $\mathbf{K_{ij}}$ along rows i and columns j of the image are estimated from the gyroscope measurements. *(b)* Next, a blurriness map $\mathbf{W}$ is generated by measuring the spatial extent of the respective kernels. *(c)* Then, individual patches are restored from multiple blurry input patches of $\mathbf{B_i}$ using natural image priors. *(d)* Finally, the sharp output $\mathbf{I}$ is assembled from the deblurred patches.

We build our blur removal algorithm on the following three observations: 1) The blurry image of a point light source in the scene (such as distant street lights at night) gives the motion blur kernel at that point of the image. 2) Smartphones today also provide a variety of sensors such as gyroscopes and accelerometers that allow reconstructing the full camera motion during camera exposure thereby giving information about the blur process. 3) Information from multiple degraded images of the same scene allows restoring a sharper image with more visible details.

Contributions

Based on the above observations we propose a new fast blur removal algorithm for unmodified smartphones by using subsequent frames from the camera preview stream combined with the built-in inertial sensors (see Figure 1). We assume a static scene and that blur is mainly caused by rotational motion of the camera. The motion of the camera during the shot is reconstructed from the gyroscope measurements, which requires time synchronization of the camera and the inertial sensors. The information from multiple subsequent frames with different blur is exploited to reconstruct a sharp image of the scene. To the best of our knowledge, this is the first application that combines blur kernel estimation from inertial sensors, patch-based non-uniform deblurring, multiframe deconvolution with natural image priors, and correction of rolling shutter deformation for unmodified smartphones. The runtime of our algorithm is in the order of a few seconds for typical preview frame size of 720×480 pixels.

2 RELATED WORK

The use of multiple blurry/noisy images for restoring a single sharp frame (without utilizing sensor data) has been proposed by Rav-Acha and Peleg [Rav05], applied for sharp panorama generation by Lie et al. [Li10], and recently for HDR and low-light photography by Ito et al [Ito14]. Combining camera frames and inertial measurement units (IMU) has been successfully used for video stabilization [For10, Han11, Kar11, Bel14], for denoising [Ito14, Rin14] and also for blur removal [Jos10, Par14, Sin14b].

Joshi et al. [Jos10] presented the first IMU-based deblurring algorithm with a DSLR camera and external gyroscope and accelerometer. In their approach, the camera and the sensors are precisely synchronized through the flash trigger. The DSLR camera has a global shutter which makes the problem easier to handle than the case of smartphones. They assume a constant uniform scene depth, which they find together with the sharp image by solving a complex optimization problem. Bae et al. [Bae13] extend this method to depth-dependent blur by attaching a depth sensor to the camera. Ringaby and Forssen [Rin14] develop a virtual tripod for smartphones by taking a series of noisy photographs, aligning them using gyroscope data, and averaging them to get a clear image, but not targeting blur removal. Köhler [Koh12] and Whyte [Why12] show in their single-frame deblurring experiments that three rotations are enough to model real camera shakes well.

Karpenko [Kar11], Ringaby [For10], and Bell [Bel14] developed methods for video stabilization and rolling shutter correction specifically for smartphones. An important issue in smartphone video stabilization is the synchronization of the camera and the IMU data because the mobile operating systems do not provide precise timestamps. Existing methods estimate the unknown parameters (time delay, frame rate, rolling shutter fill time, gyroscope drift) from a sequence of images off-line via optimization. We have found that the camera parameters might even change over time, for example the smartphones automatically adjust the frame rate depending on whether we capture a bright or a dark scene. This is an important issue because

it means we require an online calibration method. Jia and Evans [Jia14] proposed such an online camera-gyroscope calibration method for smartphones based on an extended Kalman filter (EKF). The method tracks point features over a sequence of frames and estimates the time delay, the rolling shutter fill rate, the gyroscope drift, the physical sensor offset, and even the camera intrinsics. However, it requires clean frames for feature tracking.

Recently, Park and Levoy [Par14] compared the effectiveness of jointly deconvolving multiple images degraded by small blur versus deconvolving a single image degraded by large blur, and versus averaging a set of blur-free but noisy images. They record a series of images together with gyroscope measurements on a modified tablet with advanced camera controls (e.g., exposure control, RAW data access) through the FCam API [Par11], and attach an external 750Hz gyroscope to the tablet. They estimate the rotation axis, the gyroscope drift, and the time delay between the frames and the gyroscope measurements in a non-linear optimization scheme over multiple image segments. Their optimization scheme is based on the fact that applying two different kernels to an image patch is commutative. This means in the case of true parameters, applying the generated kernels in different order results in the same blurry patch. The rolling shutter parameter is calculated off-line with the method of Karpenko [Kar11]. They report the runtime of the algorithm to be 24.5 seconds using the Wiener filter and 20 minutes using a sparsity prior for deconvolution, not mentioning whether on the tablet or on a PC.

Closest to our system is the series of work by Sindelar et al. [Sin13, Sin14a, Sin14b] who also reconstruct the blur kernels from the sensor readings of an unmodified smartphone in order to make the deconvolution non-blind. The first approach [Sin13] considers only x and y rotations and generates a single kernel as weighted line segments using Bresenham's line drawing algorithm. Unfortunately, their time calibration method is not portable to different phones. They find the exact beginning of the exposure by inspecting the logging output of the camera driver, which might be different for each smartphone model. They read the exposure time from the EXIF tag of the captured photo, however, this information is not available for the preview frames we intend to use for a live deblurring system. Extending this work in [Sin14b] the authors generate piecewise uniform blur kernels and deblur overlapping patches of the image using the Wiener filter. They also account for rolling shutter by shifting the time window of gyroscope measurements when generating blur kernels for different rows of the image. The main difference in our approach is the use of multiple subsequent images and non-blind deblurring with natural image priors.

3 BLUR MODEL

The traditional convolution model for uniform blur is written in matrix-vector form as

$$\vec{\mathbf{B}} = \mathbf{A}\vec{\mathbf{I}} + \vec{\mathbf{N}} \tag{1}$$

where $\vec{\mathbf{B}}$,$\vec{\mathbf{I}}$,$\vec{\mathbf{N}}$ denote the vectorized blurry image, sharp image, and noise term, respectively, and $\mathbf{A}$ is the sparse blur matrix. Camera shake causes non-uniform blur over the image, i.e., different parts of the image are blurred differently. We assume piecewise uniform blur and use different uniform kernels for each image region which is a good compromise between model accuracy and model complexity.

The blur kernels across the image can be found by reconstructing the camera movement, which is a path in the six-dimensional space of 3 rotations and 3 translations. A point in this space corresponds to a particular camera pose, and a trajectory in this space corresponds to the camera movement during the exposure. From the motion of the camera and the depth of the scene, the blur kernel at any image point can be derived. In this paper, we target unmodified smartphones without depth sensors so we need to make further assumptions about the scene and the motion.

Similar to Joshi et al. [Jos10], we assume the scene to be planar (or sufficiently far away from the camera) so that the blurred image can be modeled as a sum of transformations of a sharp image. The transformations of a planar scene due to camera movements can be described by a time-dependent homography matrix $\mathbf{H_t} \in \mathbb{R}^{3\times3}$. We apply the pinhole camera model with square pixels and zero skew for which the intrinsics matrix $\mathbf{K}$ contains the focal length f and the principal point $[c_x, c_y]^T$ of the camera.

Given the rotation matrix $\mathbf{R}_t$ and the translation vector $\mathbf{T_t}$ of the camera at a given time t, the homography matrix is defined as

$$\mathbf{H}_t(d) = \mathbf{K}(\mathbf{R}_t + \frac{1}{d}\mathbf{T}_t\vec{n}^T)\mathbf{K}^{-1} \tag{2}$$

where $\vec{n}$ is the normal vector of the latent image plane and d is the distance of the image plane from the camera center. The homography $\mathbf{H}_t(d)$ maps homogenous pixel coordinates from the latent image $\mathbf{I}_0$ to the transformed image $\mathbf{I}_t$ at time t:

$$\mathbf{I}_t\left((u_t, v_t, 1)^T\right) = \mathbf{I}_0\left(\mathbf{H}_t(d)(u_0, v_0, 1)^T\right) \tag{3}$$

The transformed coordinates in general are not integer valued, so the pixel value has to be calculated via bilinear interpolation, which can also be rewritten as a matrix multiplication of a sparse sampling matrix $\mathbf{A}_t(d)$ with the latent sharp image $\vec{\mathbf{I}}_0$ as $\vec{\mathbf{I}}_t = \mathbf{A}_t(d)\vec{\mathbf{I}}_0$ in vector

form. Then, we can describe the blurry image as the integration of all transformed images during the exposure time plus noise:

$$\vec{\mathbf{B}} = \int_{t_{open}}^{t_{close}} \mathbf{A}_t \cdot \vec{\mathbf{I}} dt + \vec{\mathbf{N}} = \mathbf{A} \cdot \vec{\mathbf{I}} + \vec{\mathbf{N}} \tag{4}$$

Note how this expression resembles the form of (1). While for this formula the depth of the scene is required, a common simplification is to assume zero translation [Kar11, Han11, For10] because rotation has a significantly larger impact on shake blur [Koh12, Why12, Bel14]. With only rotational motion, equation 2 is no longer dependent on the depth:

$$\mathbf{H}_t = \mathbf{K}\mathbf{R}_t\mathbf{K}^{-1} \tag{5}$$

There are also other reasons why we consider only rotational motion in our application. The smartphone's accelerometer measurements include gravity and are contaminated by noise. The acceleration values need to be integrated twice to get the translation of the camera and so the amplified noise may lead to large errors in kernel estimation.

Handling pixel-wise spatially varying blur is computationally too complex to perform on a smartphone, so we adopt a semi-non-uniform approach. We split the images into $R \times C$ overlapping regions (R and C are chosen so that we have regions of size 30×30 pixels) where we assume uniform blur and handle these regions separately. We reconstruct the motion of the camera during the exposure time from the gyroscope measurements and from the motion we reconstruct the blur kernels for each image region by transforming the image of a point light source with the above formulas. Once we have the blur kernels, fast uniform deblurring algorithms can be applied in each region, and the final result can be reassembled from the deblurred regions (possibly originating from different input frames). An overview of our whole pipeline is illustrated in Figure 2.

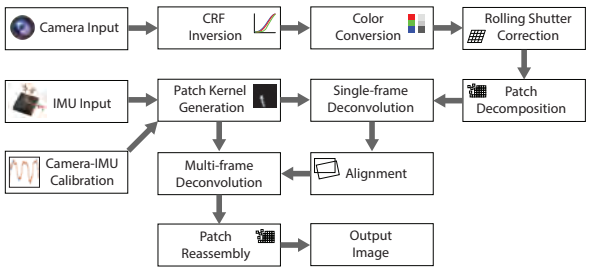

Figure 2: Overview of our blur removal pipeline

3.1 Camera-IMU calibration

Reconstructing the motion of the mobile phone during camera exposure of a given frame i with timestamp t_i requires the exact time window of sensor measurements during that frame. This is challenging to find on unmodified smartphones given that current smartphone APIs allow rather limited hardware control. We denote with t_d the delay between the recorded timestamps of sensor measurements and camera frames which we estimate prior to deblurring.

In rolling shutter (RS) cameras, the pixel values are read out row-wise from top to bottom which means 'top' pixels in an image will be transformed with 'earlier' motion than 'bottom' pixels, which has to be taken into account in our model. For an image pixel $u = [u_x, u_y]^T$ in frame i the start of the exposure is modeled as

$$t([u_x, u_y]^T, i) = t_i + t_d + t_r \frac{u_y}{h} \tag{6}$$

where t_r is the readout time for one frame and h is the total number of rows in one frame. The gyro-camera delay t_d is estimated for the first row of the frame, and the other rows are shifted in time within the range $[0, t_r]$. We set the time of each image region to the time of the center pixel in the region.

To find the unknown constants of our model, we apply once the Extended Kalman Filter (EKF)-based online gyro-camera calibration method of Jia and Evans [Jia13, Jia14]. This method estimates the rolling shutter parameter t_r, the camera intrinsics f, c_x, c_y, the relative orientation of the camera and IMU, the gyroscope bias, and even the time delay t_d. The intrinsics do not change in our application, and the relative orientation is not important because we are only interested in rotation changes which are the same in both coordinate systems. The gyroscope bias is a small and varying additive term on the measured rotational velocities which slightly influences the kernel estimation when integrated over time. However, for kernel estimation we consider only rotation changes during single camera frames, and in such short time intervals the effect of the bias can be neglected. For example, in case of a 30 Hz camera and a 200 Hz gyroscope we integrate only $200/30 \approx 6$ values during a single frame. We perform the online calibration once at the beginning of our image sequences and we assume the above parameters to be constant for the time of capturing the frames we intend to deblur. The EKF is initialized with the intrinsic values given by the camera calibration method in OpenCV.

The time delay t_d was found to slightly vary over longer sequences, so after an initial guess from the EKF, we continuously re-estimate t_d in a background thread. We continuously calculate the mean pixel shift induced by the movement measured by the gyroscope, and we also observe the mean pixel shifts in the images. The current t_d is found by correlating the curves in a sliding time window.

The last parameter, the exposure time $t_e = t_{close} - t_{open}$ of a frame can be read from the EXIF tag of JPEG images like in [Sin13], but an EXIF tag is not available for live video frames. Therefore, we lock the camera settings at the beginning and capture the first frame as a single image.

3.2 Kernel estimation from gyroscope measurements

We generate a synthetic blur kernel at a given point in the image by replaying the camera motion with a virtual camera that is looking at a virtual point light source. For any pixel $u = [u_x, u_y]$ in the image, we place the point light source to $U = [(u_x - c_x)\frac{d}{f}, (u_y - c_y)\frac{d}{f}, d]$ in 3D space. Note that the value of d is irrelevant if we consider rotations only.

First, we need to rotate all sensor samples into a common coordinate system because the raw values are measured relative to the current camera pose. The chosen reference pose is the pose at the shutter opening. Next, the measured angular velocities need to be integrated to get rotations. As described in section 3.1, we neglect the effects of gyroscope bias within the short time of the exposure. In order to get a continuous rendered kernel, we super-resolve the time between discrete camera poses where measurements exist, using spherical linear interpolation (SLERP). The transformed images of the point light source are blended together with bilinear interpolation and the resulting kernel is normalized to sum to 1. Finally, we crop the kernel to its bounding square in order to reduce the computational effort in the later deblurring step.

3.3 Camera response function

The non-linear camera response function (CRF) that converts the scene irradiance to pixel intensities has a significant impact on deblurring algorithms [Tai13]. The reason why manufacturers apply a non-linear function is to compensate the non-linearities of the human eye and to enhance the look of the image. The CRF is different for each camera model and even for different capture modes of the same camera [Xio12]. Some manufacturers disclose their CRFs but for the wide variety of smartphones only few data is available. The CRF of digital cameras can be calibrated for example using exposure bracketing [Deb97] but the current widespread smartphone APIs do not allow exposure control. The iOS 6, the upcoming Android 5.0, and custom APIs such as the FCam [Par11] expose more control over the camera but are only available for a very limited set of devices. To overcome this limitation, we follow the approach of [Li10] and assume the CRF to be a simple gamma curve with exponent 2.2. While this indeed improves the quality of our deblurred images, an online photometric calibration algorithm that tracks the automatic capture settings remains an open question.

4 SINGLE-FRAME BLUR REMOVAL

Given the piecewise uniform blur kernels, we apply the fast non-blind uniform deconvolution method of Krishnan and Fergus [Kri09] on each individual input patch to produce sharper estimates (see Figure 3). The algorithm enforces a hyper-Laplacian distribution on the gradients in the sharp image, which has been shown to be a good natural image prior [Lev09]. Assuming the image has N pixels in total, the algorithm solves for the image $\mathbf{I}$ that minimizes the following energy function:

$$\arg\min_{\mathbf{I}} \sum_{i=1}^{N} \frac{\lambda}{2}(\mathbf{I}*k-\mathbf{B})_i^2 + |(\mathbf{I}*f_x)_i|^\alpha + |(\mathbf{I}*f_y)_i|^\alpha \quad (7)$$

where k is the kernel, and $f_x = [1\ -1]$ and $f_y = [1\ -1]^T$ denote differential operators in horizontal and vertical direction, respectively. λ is a balance factor between the data and the prior terms. The notation $F_i^d\mathbf{I} := (\mathbf{I} * f_d)_i$ and $K_i\mathbf{I} := (\mathbf{I} * k)_i$ will be used in the following for brevity. Introducing auxiliary variables w_i^x and w_i^y (together denoted as $\mathbf{w}$) at each pixel i allows moving the $F_i^d\mathbf{I}$ terms outside the $|\cdot|^\alpha$ expression, thereby separating the problem into two sub-problems.

$$\begin{aligned}\arg\min_{\mathbf{I},\mathbf{w}} \sum_{i=1}^{N} \frac{\lambda}{2}(K_i\mathbf{I}-\mathbf{B}_i)^2 + \frac{\beta}{2}\|F_i^x\mathbf{I}-w_i^x\|_2^2 \ + \\ + \ \frac{\beta}{2}\|F_i^y\mathbf{I}-w_y^2\|_2^2 + |w_i^x|^\alpha + |w_i^y|^\alpha\end{aligned} \quad (8)$$

The β parameter enforces the solution of eq. 8 to converge to the solution of eq. 7, and its value is increased in every iteration. Minimizing eq. 8 for a fixed β can be done by alternating between solving for $\mathbf{I}$ with fixed $\mathbf{w}$ and solving for $\mathbf{w}$ with fixed $\mathbf{I}$. The first sub-problem is quadratic, which makes it simple to solve in the Fourier domain. The second sub-problem is pixel-wise separable, which is trivial to parallelize. Additionally, for certain values of α an analytical solution of the w-subproblem can be found, especially for $\alpha = \frac{1}{2},\ \frac{2}{3},\ 1$, and for other values a Newton-Raphson root finder can be applied. We experimentally found that $\alpha = \frac{1}{2}$ gives the best results. For further details of the algorithm please refer to [Kri09]. For smoothing discontinuities that may produce ringing artifacts, we perform edge tapering on the overlapping regions before deblurring.

5 MULTI-FRAME BLUR REMOVAL

One of the main novelties of our method is to aid the restoration of a single frame $\mathbf{B}$ with information from preceding and/or subsequent degraded frames $\mathbf{B}_j, 1 \leq j \leq M$ from the camera stream. We first undistort each input frame using the RS-rectification method of

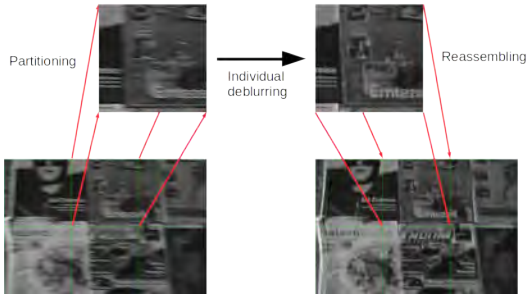

Figure 3: Illustration of our piecewise uniform deblurring method with $R = 2$ rows and $C = 3$ columns. The regions are deblurred independently with the respective kernels and afterwards reassembled in the final image.

Karpenko et al. [Kar11] which is a simple texture warping step on the GPU. We initialize the warping with the same gyroscope data that is used for kernel estimation. Next, we perform single-frame deblurring on every $\mathbf{B}_j$ as well as $\mathbf{B}$ to get sharper estimates $\tilde{\mathbf{I}}_j$ and $\tilde{\mathbf{I}}$, respectively.

After deblurring the single frames, we align all sharpened estimates with $\tilde{\mathbf{I}}$. For each $\tilde{\mathbf{I}}_j$, we calculate the planar homography $\mathbf{H}_j$ that maps $\tilde{\mathbf{I}}_j$ to $\tilde{\mathbf{I}}$. To find the homographies, we perform basic SURF [Bay08] feature matching between each estimate $\tilde{\mathbf{I}}_j$ and $\tilde{\mathbf{I}}$ in a RANSAC loop. Each homography is calculated from the inlier matches such that the reprojection error is minimized. We found that this method is robust even in the case of poorly restored estimates (e.g., in case of large blurs); however, the homography matching can fail if there are many similar features in the recorded scene. Frames that fail the alignment step are discarded.

Finally, we patch-wise apply the warped estimates $\mathbf{I}_j$ as additional constraints on $\mathbf{I}$ in our modified deblurring algorithm. We add a new penalty term γ_{multi} to equation 7 which describes the deviation of the latent image $\mathbf{I}$ from the M other estimates:

$$\gamma_{multi} = \frac{\mu}{2\sum_{j=1}^{M}\mu_j}\sum_{j=1}^{M}\mu_j||\mathbf{I}-\mathbf{I}_j||_2^2 \qquad (9)$$

The weights μ_j are chosen inversely proportional to the 'blurriness' of the corresponding patch in image $\mathbf{B}_j$. The blurriness is defined as the standard deviation (spatial extent) of the blur kernel in the patch. Note that the weights are recalculated for every patch in our non-uniform deblurring approach. The benefit of calculating the weights for each patch independently from the rest of the image is that we can both *spatially* and *temporally* give more weight to sharper patches in the input stack of images. This is advantageous if the same part of the scene got blurred differently in subsequent images. An example colormap of the weights (W) is visualized in Figure 1 where each input image is represented as a distinct color and W shows how much the different input patches influence an output tile.

Analog to the single-frame case, we proceed with the half-quadratic penalty method to separate the problem into $\mathbf{I}$ and $\mathbf{w}$ sub-problems. In the extended algorithm, we only need to calculate one additional Fourier transform in each iteration which keeps our multi-frame method fast. The step-by-step derivation of the solution can be found in the supplemental material.

6 EXPERIMENTS

We have implemented the proposed algorithm in OpenCV[1] and the warping in OpenGL ES 2.0[2] which makes it portable between a PC and a smartphone. We recorded grayscale camera preview sequences of resolution 720×480 at 30 Hz together with gyroscope data at 200 Hz on a Google Nexus 4 smartphone and we performed the following experiments on a PC. The gyro-camera calibration is performed on the first few hundred frames with Jia's implementation [Jia14] in Matlab.

6.1 Kernel generation

To test the accuracy of kernel generation, we recorded a sequence in front of a point light grid, and also generated the kernels from the corresponding gyroscope data. Ideally, the recorded image and the generated image should look identical, and after (single-frame) deblurring using the generated kernels the resulting image should show a point light grid again. Figure 4 illustrates that our kernel estimates are close to the true kernels, however, the bottom part is not matching perfectly because of residual errors in the online calibration.

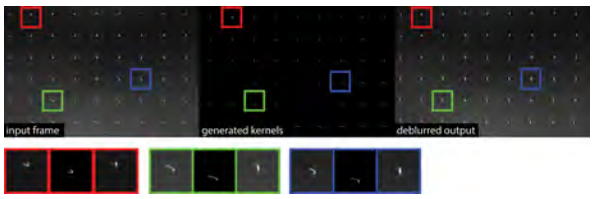

Figure 4: Kernel generation example. Left: blurry photo of a point grid. Middle: kernels estimated from gyroscope data. Right: the deblurred image is close to a point grid again. (Please zoom in for viewing)

6.2 Removing synthetic blur

Starting from a sharp image and previously recorded gyroscope measurements of deliberate handshake, we generated a set of 5 blurry images as input. In our restoration algorithm, we split the 720×480 input images to a grid of $R \times C = 24 \times 16$ regions and for each

[1] http://www.opencv.org/
[2] https://www.khronos.org/opengles/

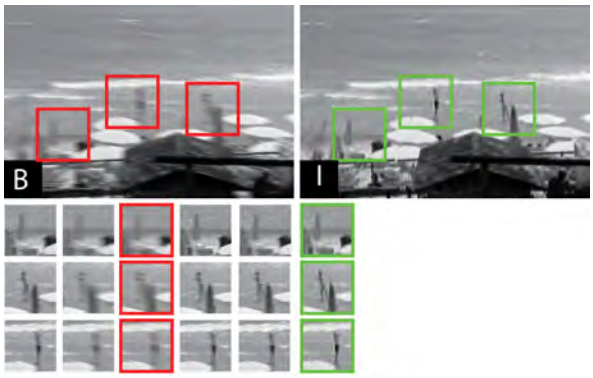

Figure 5: Removing synthetic blur. B is the main input image and 4 neighboring images aid the blur removal, I is our result. Bottom: 3 corresponding patches from the 5 input images, and from the result.

region we individually generate the blur kernel from the gyroscope data. Our multi-frame deconvolution result is shown in Figure 5. The runtime of our algorithm on 5 input images is 18 seconds on a laptop with a 2.40 GHz Core i7-4700MQ CPU (without calibration time).

Next, we test different deconvolution algorithms in our piecewise uniform blur removal for restoring a single frame. We test the Wiener filter, the Richardson-Lucy algorithm, and Krishnan's algorithm as our deconvolution step. For comparison, we also show the results of Photoshop's ShakeReduction feature which is a uniform blind deconvolution method (i.e., can not make use of our gyro-generated kernels). The quality metrics PSNR (peak signal to noise ratio) and SSIM (structure similarity index [Wan04]) of the images are listed in Table 1. The Wiener filter (W) is the fastest method for

	B	W	RL	KS	KM	PS
PSNR	22.374	20.613	23.207	22.999	**24.215**	21.914
SSIM	0.616	0.534	0.657	0.621	**0.673**	0.603

Table 1: Quantitative comparison of various deconvolution steps in our framework. Blurry input image (B), Wiener filter (W), Richardson-Lucy (RL), single-frame Krishnan (KS), multi-frame Krishnan (KM) (3 frames), Photoshop ShakeReduction (PS) (blind uniform deconvolution)

patch-wise single-frame deblurring but produces ringing artifacts that even lower the quality metrics. The output of the Richardson-Lucy algorithm (RL) achieved higher score but surprisingly remained blurry, while Krishnan's algorithm (KS) tends to smooth the image too much. We think this might stem from the fact that the small 30×30 regions do not contain enough image gradients to steer the algorithm to the correct solution. However, Krishnan's algorithm with our extension to multiple input regions (KM) performs the best. Photoshop's ShakeReduction algorithm assumes uniform blur [Cho09] so while it restored the bottom part of the image correctly, the people in the middle of the image remained blurry. The images in higher resolution can be found in the supplement.

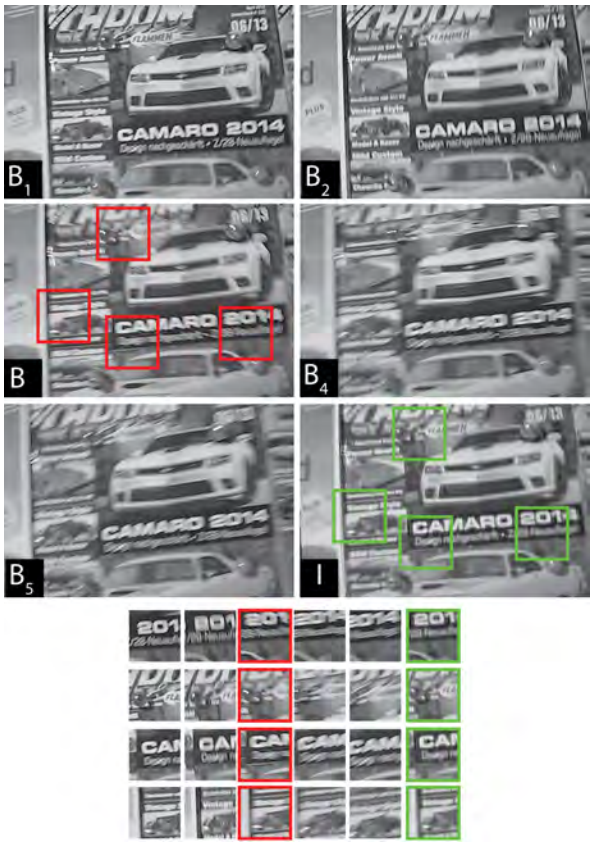

Figure 6: Restoring B with the help of $B_{1,2,4,5}$, all degraded with real motion blur. We also added four marble balls to the scene that act as point light sources and show the true blur kernels at their locations.

6.3 Removing real blur

Figure 6 shows the results of a real example with a static planar scene close to the camera. We used 5 input images degraded by real motion blur. Selected patches illustrate how our algorithm restores different parts of the image by locally steering the deconvolution towards the sharper input tiles. Note, however, that in the second selected patch some details are missing that were present in the sharpest of the input patches. This is because our algorithm does not directly copy that patch from B_1 but applies it within the deconvolution of the corresponding patch in B. A slight misalignment of the five input tiles leads to smoother edges in the multi-frame deconvolution result. The misalignment may stem from the homography estimation which is prone to errors if the single-frame deconvolution is imperfect or from the rolling shutter rectification which is only an approximation of the true image distortion. Figure 7 shows another example of restoring a sharp image from three blurry ones.

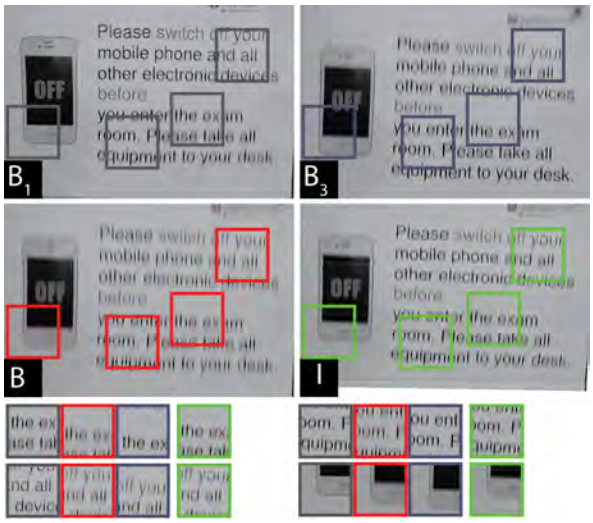

Figure 7: Restoring B with the help of $B_{1,3}$, all degraded with real motion blur. The original misalignment is kept and the rectified result is left uncropped for visualization. (High-resolution images are in the supplement)

6.4 Discussion and limitations

As shown in the experiments, the proposed algorithm is able to restore non-uniformly blurred images. However, there are limitations and assumptions we need to keep in mind. Our non-blind deconvolution step assumes a perfect kernel estimate so a good calibration is crucial for success. The selected gyro-camera calibration method is sensitive to the initialization values which are not trivial to find for different smartphone models. We need to assume that a short sequence with detectable feature tracks for calibration before deblurring exists. However, the calibration does not need to be done every time but only when the camera settings change. We expect that future camera and sensor APIs like StreamInput[3] will provide better synchronization capabilities that allow precise calibration.

Our blur model can generate rotational motion blur kernels at any point of the image, but the model is incorrect if the camera undergoes significant translation or if objects are moving in the scene during the exposure time. In multiframe deblurring, the image alignment based on feature matching might fail if the initial deblurring results are wrong. As we also know the camera motion between the frames, image alignment could be done with the aid of sensor data instead of pure feature matching but then the gyroscope bias needs to be compensated. The restriction to planar scenes and rotational motion overcomes the necessity of estimating the depth of the scene at each pixel.

Our algorithm was formulated for grayscale images. Extending it to color images would be possible by solving the grayscale problem for the RGB color channels separately, however, the color optimizations of the smartphone driver may introduce different non-linear CRFs for each channel, which needs to be handled carefully. The calibration of the per-channel CRFs using standard methods will become possible with the upcoming smartphone APIs that allow low-level exposure control.

The runtime of the algorithm depends mainly on the size of the input images. In fact, we perform $R \times C \times M$ non-blind deconvolutions on patches but as the patches are overlapping, we process somewhat more pixels than the image contains. Our tests were conducted off-line on a PC but each component of our algorithm is portable to a smartphone with little modifications.

[3] https://www.khronos.org/streaminput/

7 CONCLUSION

We proposed a new algorithm for unmodified off-the-shelf smartphones for the removal of handshake blur from photographs of planar surfaces such as posters, advertisements, or price tags. We re-formulated the fast non-blind uniform blur removal algorithm of Krishnan and Fergus [Kri09] to multiple input frames and to non-uniform blur. We rendered piecewise uniform blur kernels from the gyroscope measurements and we adaptively weighted the input patches in a multiframe deconvolution framework based on their blurriness. The distortion effects of the rolling shutter of the smartphone camera were compensated prior to deblurring. We applied existing off-line and on-line methods for gyroscope and camera calibration, however, a robust on-line calibration method is still an open question. We have shown the effectiveness of our method in qualitative experiments on images degraded by synthetic and real blur. Our future work will focus on fast blind deblurring that is initialized with our rendered motion blur kernels so that less iterations are required in the traditional multiscale kernel estimation.

ACKNOWLEDGEMENTS

We thank Fabrizio Pece and Tobias Nägeli for the productive discussions and the anonymous reviewers for their comments and suggestions.

8 REFERENCES

[Bae13] H. Bae, C. C. Fowlkes, and P. H. Chou. Accurate motion deblurring using camera motion tracking and scene depth. In *IEEE Workshop on Applications of Computer Vision (WACV)*, 2013.

[Bay08] H. Bay, T. Tuytelaars, and L. van Gool. SURF: Speeded Up Robust Features. In *European Conference on Computer Vision (ECCV)*, 2006.

[Bel14] S. Bell, A. Troccoli, and K. Pulli. A non-linear filter for gyroscope-based video stabilization. In *European Conference on Computer Vision (ECCV)*, 2014.

[Cho09] S. Cho and S. Lee. Fast motion deblurring. In *ACM SIGGRAPH Asia*, 2009.

[Deb97] P. E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *ACM SIGGRAPH*, 1997.

[For10] P.-E. Forssen and E. Ringaby. Rectifying rolling shutter video from hand-held devices. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[Gol12] A. Goldstein and R. Fattal. Blur-kernel estimation from spectral irregularities. In *European Conference on Computer Vision (ECCV)*, 2012.

[Han11] G. Hanning, N. Forslow, P.-E. Forssen, E. Ringaby, D. Tornqvist, and J. Callmer. Stabilizing cell phone video using inertial measurement sensors. In *IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2011.

[Ito14] A. Ito, A. C. Sankaranarayanan, A. Veeraraghavan, and R. G. Baraniuk. BlurBurst: Removing blur due to camera shake using multiple images. In *ACM Transactions on Graphics*, 2014.

[Jia13] C. Jia and B. Evans. Online calibration and synchronization of cellphone camera and gyroscope. In *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2013.

[Jia14] C. Jia and B. Evans. Online camera-gyroscope autocalibration for cell phones. In *IEEE Transactions on Image Processing*, 23(12), 2014.

[Jos10] N. Joshi, S. B. Kang, C. L. Zitnick, and R. Szeliski. Image deblurring using inertial measurement sensors. In *ACM SIGGRAPH*, 2010.

[Jos08] N. Joshi, R. Szeliski, and D. Kriegman. Psf estimation using sharp edge prediction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[Kar11] A. Karpenko, D. Jacobs, J. Baek, and M. Levoy. Digital video stabilization and rolling shutter correction using gyroscopes. Technical report, Stanford University, 2011.

[Koh12] R. Köhler, M. Hirsch, B. Mohler, B. Schölkopf, and S. Harmeling. Recording and playback of camera shake: Benchmarking blind deconvolution with a real-world database. In *European Conference on Computer Vision (ECCV)*, 2012.

[Kri09] D. Krishnan and R. Fergus. Fast image deconvolution using hyper-Laplacian priors. In *Advances in Neural Information Processing Systems (NIPS)*. 2009.

[Lev09] A. Levin, Y. Weiss, F. Durand, and W. Freeman. Understanding and evaluating blind deconvolution algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[Li10] Y. Li, S. B. Kang, N. Joshi, S. Seitz, and D. Huttenlocher. Generating sharp panoramas from motion-blurred videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[Par14] S. Park and M. Levoy. Gyro-based multi-image deconvolution for removing handshake blur. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[Par11] S. H. Park, A. Adams, and E.-V. Talvala. The FCam API for programmable cameras. In *ACM International Conference on Multimedia (MM)*, 2011.

[Rav05] A. Rav-Acha and S. Peleg. Two motion-blurred images are better than one. *Pattern Recognition Letters*, 26(3), 2005.

[Rin14] E. Ringaby and P.-E. Forssen. A virtual tripod for hand-held video stacking on smartphones. In *IEEE International Conference on Computational Photography (ICCP)*, 2014.

[Sin13] O. Sindelar and F. Sroubek. Image deblurring in smartphone devices using built-in inertial measurement sensors. In *Journal of Electronic Imaging*, 22(1), 2013.

[Sin14a] O. Sindelar, F. Sroubek, and P. Milanfar. A smartphone application for removing handshake blur and compensating rolling shutter. In *IEEE Conference on Image Processing (ICIP)*, 2014.

[Sin14b] O. Sindelar, F. Sroubek, and P. Milanfar. Space-variant image deblurring on smartphones using inertial sensors. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2014.

[Sun13] L. Sun, S. Cho, J. Wang, and J. Hays. Edge-based blur kernel estimation using patch priors. In *IEEE International Conference on Computational Photography (ICCP)*, 2013.

[Tai13] Y.-W. Tai, X. Chen, S. Kim, S. J. Kim, F. Li, J. Yang, J. Yu, Y. Matsushita, and M. Brown. Nonlinear camera response functions and image deblurring: Theoretical analysis and practice. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(10), 2013.

[Tai10] Y.-W. Tai, H. Du, M. Brown, and S. Lin. Correction of spatially varying image and video motion blur using a hybrid camera. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6), 2010.

[Wan04] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. In *IEEE Transactions on Image Processing*, 13(4), 2004.

[Why12] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce. Non-uniform deblurring for shaken images. In *International Journal of Computer Vision*, 98(2), 2012.

[Xio12] Y. Xiong, K. Saenko, T. Darrell, and T. Zickler. From pixels to physics: Probabilistic color de-rendering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[Xu13] L. Xu, S. Zheng, and J. Jia. Unnatural L0 sparse representation for natural image deblurring. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.