Image Deblurring using Inertial Measurement Sensors

Neel Joshi Sing Bing Kang C. Lawrence Zitnick Richard Szeliski
Microsoft Research

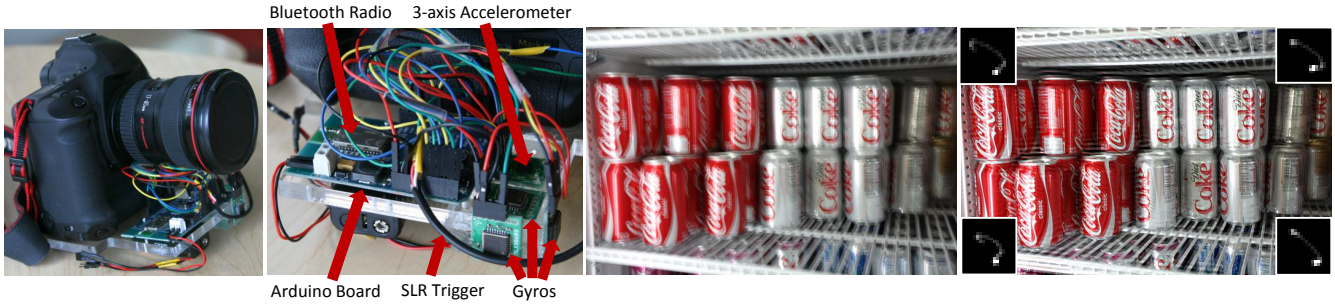

Figure 1: *An SLR Camera instrumented with our image deblurring attachment that uses inertial measurement sensors and the input image in an "aided blind-deconvolution" algorithm to automatically deblur images with spatially-varying blurs (first two images). A blurry input image (third image) and the result of our method (fourth image). The blur kernel at each corner of the image is shown at* $2\times$ *size.*

Abstract

We present a deblurring algorithm that uses a hardware attachment coupled with a natural image prior to deblur images from consumer cameras. Our approach uses a combination of inexpensive gyroscopes and accelerometers in an energy optimization framework to estimate a blur function from the camera's acceleration and angular velocity during an exposure. We solve for the camera motion at a high sampling rate *during* an exposure and infer the latent image using a joint optimization. Our method is completely automatic, handles per-pixel, spatially-varying blur, and out-performs the current leading image-based methods. Our experiments show that it handles large kernels – up to at least 100 pixels, with a typical size of 30 pixels. We also present a method to perform "ground-truth" measurements of camera motion blur. We use this method to validate our hardware and deconvolution approach. To the best of our knowledge, this is the first work that uses 6 DOF inertial sensors for dense, per-pixel spatially-varying image deblurring and the first work to gather dense ground-truth measurements for camera-shake blur.

1 Introduction

Intentional blur can be used to great artistic effect in photography. However, in many common imaging situations, blur is a nuisance. Camera motion blur often occurs in light-limited situations and is one of the most common reason for discarding a photograph. If the blur function is known, the image can be improved by deblurring it with a non-blind deconvolution method. However, for most images, the blur function is unknown and must be recovered. Recovering both the blur or "point-spread function" (PSF) and the desired deblurred image from a single blurred input (known as the blind-deconvolution problem) is inherently ill-posed, as the observed blurred image provides only a partial constraint on the solution.

Prior knowledge about the image or kernel can disambiguate the potential solutions and make deblurring more tractable [Fergus et al. 2006]. Most current approaches use image priors modeled from local image statistics. While these approaches have shown some promise, they have some limitations: they generally assume spatially invariant blur, have long run times, cannot be run on high-resolution images, and often fail for large image blurs. One of the most significant aspects of camera-shake blur that recent work has overlooked is that the blur is usually not spatially invariant. This can be depth-dependent due to camera translation, or depth-independent, due to camera rotation. Furthermore, image-based methods cannot always distinguish unintended camera-shake blur from intentional defocus blur, e.g., when there is an intentional shallow depth of field. As many methods treat all types of blur equally, intentional defocus blur may be removed, creating an over-sharpened image.

We address some of these limitations with a combined hardware and software-based approach. We have present a novel hardware attachment that can be affixed to any consumer camera. The device uses inexpensive gyroscopes and accelerometers to measure a camera's acceleration and angular velocity during an exposure. This data is used as an input to a novel "aided blind-deconvolution" algorithm that computes the spatially-varying image blur and latent deblurred image. We derive a model that handles spatially-varying blur due to full 6-DOF camera motion and spatially-varying scene depth; however, our system assumes spatially invariant depth.

By instrumenting a camera with inertial measurement sensors, we can obtain relevant information about the camera motion and thus the camera-shake blur; however, there are many challenges in using this information effectively. Motion tracking using inertial sensors is prone to significant error when tracking over an extended period of time. This error, known as "drift", occurs due to the integration of the noisy measurements, which leads to increasing inaccuracy in the tracked position over time. As we will show in our experiments, using inertial sensors directly is not sufficient for camera tracking and deblurring.

Instead, we use the inertial data and the recorded blurry image together with an image prior in a novel "aided blind-deconvolution" method that computes the camera-induced motion blur and the latent deblurred image using an energy minimization framework. We consider the algorithm to be "aided blind-deconvolution", since it is only given an estimate of the PSF from the sensors. Our method is completely automatic, handles per-pixel, spatially-varying blur, out-performs current leading image-based methods, and our experiments show it handles large kernels – up to 100 pixels, with a typical size of 30 pixels.

As a second contribution, we expand on previous work and develop a validation method to recover "ground-truth", per-pixel spatially-varying motion blurs due to camera-shake. We use this method to validate our hardware and blur estimation approach, and also use it to study the properties of motion blur due to camera shake.

Specifically, our work has the following contributions: (1) a novel hardware attachment for consumer cameras that measures camera motion, (2) a novel aided blind-deconvolution algorithm that combines a natural image prior with our sensor data to estimate a spatially-varying PSF, (3) a deblurring method that using a novel spatially-varying image deconvolution method to only remove the camera-shake blur and leaves intentional artistic blurs (i.e., shallow DOF) intact, and (4) a method for accurately measuring spatially-varying camera-induced motion blur.

2 Related Work

Image deblurring has recently received a lot of attention in the computer graphics and vision communities. Image deblurring is the combination of two tightly coupled sub-problems: PSF estimation and non-blind image deconvolution. These problems have been addressed both independently and jointly [Richardson 1972]. Both are longstanding problems in computer graphics, computer vision, and image processing, and thus the entire body of previous work in this area is beyond what can be covered here. For a more in depth review of earlier work in blur estimation, we refer the reader to the survey paper by Kundur and Hatzinakos [1996].

Blind deconvolution is an inherently ill-posed problem due to the loss of information during blurring. Early work in this area significantly constrained the form of the kernel, while more recently, researchers have put constraints on the underlying sharp image [Bascle et al. 1996; Fergus et al. 2006; Yuan et al. 2007; Joshi et al. 2008]. Alternative approaches are those that use additional hardware to augment a camera to aid in the blurring process [Ben-Ezra and Nayar 2004; Tai et al. 2008; Park et al. 2008].

The most common commercial approach for reducing image blur is image stabilization (IS). These methods, used in high-end lenses and now appearing in lower-end point and shoot cameras, use mechanical means to dampen camera motion by offsetting lens elements or translating the sensor. IS methods are similar to our work in that they use inertial sensors to reduce blur, but there are several significant differences. Fundamentally, IS tries to dampen motion by assuming that the past motion predicts the future motion [Canon 1993]; however, it does not counteract the actual camera motion during an exposure nor does it actively remove blur – it only reduces blur. In contrast, our method records the actual camera motion and removes the blur from the image. A further difference is that IS methods can only dampen 2D motion, e.g., these methods will not handle camera roll, while our method can handle six degrees of motion. That said, our method could be used in conjunction with image stabilization, if one were able to obtain readings of the mechanical offsetting performed by the IS system.

Recent research in hardware-based approaches to image deblurring modify the image capture process to aid in deblurring. In this area, our work is most similar to approaches that uses hybrid cameras [Ben-Ezra and Nayar 2004; Tai et al. 2008], which track camera motion using data from a video camera attached to a still camera. This work compute a global frame-to-frame motion to calculate the 2D camera motion during the image-exposure window. Our work is similar, since we also track motion during the exposure window; however, we use inexpensive, small, and lightweight sensors instead of a second camera. This allows us to measure more degrees of camera motion at a higher-rate and lower cost. Another difficulty of the hybrid camera approach that we avoid is that it can be difficult to get high-quality, properly exposed images out of the video camera in the low light conditions where image blur is prevalent. We do, however, use a modified form of Ben-Ezra and Nayar's work in a controlled situation to help validate our estimated camera motions.

Also similar to our work is that of Park et al. [2008], who use a 3-axis accelerometer to measure motion blur. The main difference between our work and theirs is that we additionally measure 3 axes of rotational velocity. As we discuss later, we have found 3 axes of acceleration insufficient for accurately measuring motion blur, as rotation is commonly a significant part of the blur.

Our work is complementary to the hardware-based deblurring work of Levin el al.'s [2008], who show that by moving a camera along a parabolic arc, one can create an image such that 1D blur due to objects in the scene can be removed regardless of the speed or direction of motion. Our work is also complementary to that of Raskar et al. [2006], who developed a fluttered camera shutter to create images with blur that was more easily inverted.

3 Deblurring using Inertial Sensors

In this section, we describe the design challenges and decisions for building our sensing platform for image deblurring. We first review the image blur process from the perspective of the six degree motion of a camera. Next we give an overview of camera dynamics and inertial sensors followed by our deblurring approach.

3.1 Camera Motion Blur

Spatially invariant image blur is modeled as the convolution of a latent sharp image with a shift-invariant kernel plus noise, which is typically considered to be additive white Gaussian noise. Specifically, blur formation is commonly modeled as:

$$B = I \otimes K + N, \tag{1}$$

where K is the blur kernel, $N \sim \mathcal{N}(0, \sigma^2)$ is the noise. With a few exceptions, most image deblurring work assumes a spatially invariant kernel; however, this often does not hold in practice [Joshi et al. 2008; Levin et al. 2009]. In fact there are many properties of a camera and a scene that can lead to spatially-varying blur: (1) depth dependent defocus blur, (2) defocus blur due to focal length variation over the image plane, (3) depth dependent blur due to camera translation, (4) camera roll motion, and (5) camera yaw and pitch motion when there are strong perspective effects. In this work, our goal is to handle only camera induced motion blur, i.e., spatially-varying blur due to the last three factors.

First, let us consider the image a camera captures during its exposure window. The intensity of light from a scene point (X, Y, Z) at an instantaneous time t is captured on the image plane at a location (u_t, v_t), which is a function of the camera projection matrix P_t. In homogenous coordinates, this can be written as:

$$(u_t, v_t, 1)^T = P_t (X, Y, Z, 1)^T. \tag{2}$$

If there is camera motion, P_t varies with time as a function of camera rotation and translation causing fixed points in the scene to project to different locations at each time. The integration of these projected observations creates a blurred image, and the projected trajectory of each point on the image plane is that point's point-spread function (PSF). The camera projection matrix can be decomposed as:

$$P_t = K \Pi E_t, \tag{3}$$

where K is the intrinsics matrix, Π is the canonical perspective projection matrix, and E_t is the time dependent extrinsics matrix that is composed of the camera rotation R_t and translation T_t. In the case of image blur, it is not necessary to consider the absolute motion of the camera, only the relative motion and its effect on the image. We model this by considering the planar homography that maps the initial projection of points at $t = 0$ to any other time t, i.e., the reference coordinate frame is coincident with the frame at time $t = 0$:

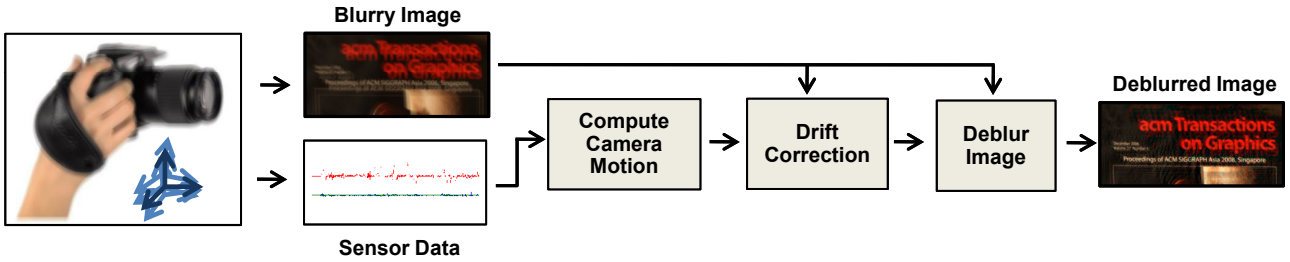

Figure 2: *Our image deblurring algorithm: First the sensor data is used to compute an initial guess for the camera motion. From this, we use the image data to search for a small perturbation of the x and y end points of the camera translation, to overcome drift. Using this result, we compute the spatially-varying blur matrix and deconvolve the image.*

$$H_t(d) = [K(R_t + \frac{1}{d}T_t N^T)K^{-1}] \tag{4}$$

$$(u_t, v_t, 1)^T = H_t(d)(u_0, v_0, 1)^T, \tag{5}$$

for a particular depth d, where N is the unit vector that is orthogonal to the image plane.

Thus given an image I at time $t = 0$, the pixel value of any subsequent image is:

$$I_t(u_t, v_t) = I(H_t(d)(u_0, v_0, 1)^T). \tag{6}$$

This image warp can be re-written in matrix form as:

$$\vec{I_t} = A_t(d)\vec{I}, \tag{7}$$

where $\vec{I_t}$ and $\vec{I}$ are column-vectorized images and $A_t(d)$ is a sparse re-sampling matrix that implements the image warping and resampling due to the homography. Each row of $A_t(d)$ contains the weights to compute the value at pixel (u_t, v_t) as the interpolation of the point $(u_0, v_0, 1)^T = H_t(d)^{-1}(u_t, v_t, 1)^T$ – we use bilinear interpolation, thus there are four values per row. We can now define an alternative formulation for image blur as the integration of applying these homographies over time:

$$\vec{B} = \int_0^s \left[A_t(d)\vec{I}dt\right]. \tag{8}$$

The spatially invariant kernel in Equation 1 is now replaced by a spatially-variant blur represented by a sparse-matrix:

$$A(d) = \int_0^s A_t(d)dt, \tag{9}$$

our spatially-varying blur model is given by:

$$\vec{B} = A(d)\vec{I} + N. \tag{10}$$

Thus, the camera-induced, spatially-varying blur estimation process is reduced to estimating the rotations R and translations T for times $[0...t]$, the scene depths d, and the camera intrinsics K. By representing the camera-shake blur in the six degrees of motion of the camera, instead of purely in the image plane, the number of unknowns is reduced significantly – there are six unknowns per each of M time-steps, an unknown depth per-pixel ($w \times h$ unknowns), and the camera intrinsics, of which the focal length is the most important factor. This results in $6M + wh + 1$ unknowns as opposed to an image-based approach that must recover an $k \times k$ kernel for each pixel, resulting in $k^2 \times wh$ unknowns. In practice, since we assume a single depth for the scene, the unknowns in our system reduce to $6M + 2$.

3.2 Spatially-Varying Deconvolution

If these values are known, the image can be deblurred using non-blind deconvolution. We modify the formulation of Levin et al. [2007] to use our spatially-varying blur model. We formulate image deconvolution using a Bayesian framework and find the most likely estimate of the sharp image I, given the observed blurred image B, the blur matrix A, and noise level σ^2 using a *maximum a posteriori* (MAP) technique.

We express this as a maximization over the probability distribution of the posterior using Bayes' rule. The result is a minimization of a sum of negative log likelihoods:

$$P(I|B, A) = P(B|I)P(I)/P(B) \tag{11}$$

$$\underset{I}{\text{argmax}}\, P(I|B) = \underset{I}{\text{argmin}}[L(B|I) + L(I)]. \tag{12}$$

The problem of deconvolution is now reduced to minimizing the negative log likelihood terms. Given the blur formation model (Equation 1), the "data" negative log likelihood is:

$$L(B|I) = ||\vec{B} - A(d)\vec{I}||^2/\sigma^2. \tag{13}$$

The contribution of our deconvolution approach is this new data term that uses the spatially-varying model derived in the previous section.

Our "image" negative log likelihood is the same as Levin et al.'s [2007] sparse gradient penalty, which enforces a hyper-Laplacian distribution: $L(I) = \lambda||\nabla I||^{0.8}$. The minimization is performed using iteratively re-weighted least-squares [Stewart 1999].

3.3 Rigid Body Dynamics and Inertial Sensors

As discussed in the previous section, camera motion blur is dependent on rotations R and translations T for times $[0...t]$, the scene depths d, and camera intrinsics K. In this section, we discuss how to recover the camera rotations and translations, and in Section 4.2 we address recovering camera intrinsics.

Any motion of a rigid body and any point on that body can be parameterized as a function of six unknowns, three for rotation and three for translation. We now describe how to recover these quantities given inertial measurements from accelerometers and gyroscopes.

Accelerometers measure the total acceleration at a given point along an axis, while gyroscopes measure the angular velocity at a given point around an axis. Note that for a moving rigid body, the pure rotation at all points is the same, while the translations for all points is not the same when the body is rotating.

Before deriving how to compute camera motion from inertial measurements, we first present our notation, as summarized in Table 1.

A rigid body, such as a camera, with a three axis accelerometer and three axis gyroscope (three accelerometers and gyroscopes mounted along x, y, and z in a single chip, respectively) measures the following accelerations and angular velocities:

$$\vec{\omega}_t^t = {}^tR^i * \vec{\omega}_t^i \tag{14}$$

$$\vec{a}_p^t = {}^tR^i \left(\vec{a}_t^i + \vec{g}^i + (\vec{\omega}_t^i \times (\vec{\omega}_t^i \times \vec{r}_p^q)) + (\vec{\alpha}_t^i \times \vec{r}_p^q)\right). \tag{15}$$

The measured acceleration is the sum of the acceleration due to translation of the camera, centripetal acceleration due to rotation,

Symbol	Description
${}^tR^i$	Initial to current frame
$\vec{\theta}_t^i, \vec{\omega}_t^i, \vec{\alpha}_t^i$	Current angular pos., vel., and accel. in initial frame
$\vec{\omega}_t^t$	**Current angular vel. in the current frame**
$\vec{x}_t^i, \vec{v}_t^i, \vec{a}_t^i$	Current pos., vel. and accel. in the initial frame
$\vec{a}_p^t$	**Accel. of the accelerometer in the current frame**
$\vec{x}_p^i$	**Position of the accelerometer in the initial frame**
$\vec{r}_p^q$	Vector from the accelerometer to center of rotation
$\vec{g}^i$	Gravity in the camera's initial coordinate frame

Table 1: *Quantities in **bold** indicate measured or observed values. Note that these are vectors, i.e., three axis quantities. The "superscript" character indicates the coordinate system of a value and the "subscript" indicates the value measured.*

the tangential component of angular acceleration, and gravity, all rotated into the current frame of the camera. The measured angular velocity is the camera's angular velocity also rotated in the current frame of the camera. To recover the relative camera rotation, it is necessary to recover the angular velocity for each time-step t in the coordinate system of the initial frame $\vec{\omega}_t^i$, which can be integrated to get the angular position. To recover relative camera translation, we need to first compute the accelerometer position for each time-step relative to the initial frame. From this, we can recover the camera translation.

The camera rotation can be recovered by sequentially integrating and rotating the measured angular velocity into the initial camera frame.

$$\vec{\theta}_t^i = ({}^iR^{t-1}\vec{\omega}_{t-1}^{t-1})\Delta t + \vec{\theta}_{t-1}^i \tag{16}$$

$${}^tR^i = angleAxisToMat(\vec{\theta}_t^i), \tag{17}$$

where "$angleAxisToMat$" converts the angular position vector to a rotation matrix. Since we are only concerned with relative rotation, the initial rotation is zero:

$$\vec{\theta}_{t=0}^i = 0, \quad {}^{t=0}R^i = Identity. \tag{18}$$

Once the rotations are computed for each time-step, we can compute the acceleration in the initial frame's coordinate system:

$$\vec{a}_p^i = {}^iR^t\vec{a}_p^t, \tag{19}$$

and integrate the acceleration, minus the constant acceleration of gravity, to get the accelerometer's relative position at each time-step:

$$\vec{v}_p^i(t) = (\vec{a}_p^i(t-1) - \vec{g}^i)\Delta t + \vec{v}_p^i(t-1) \tag{20}$$

$$\vec{x}_p^i(t) = 0.5 * (\vec{a}_p^i(t-1) - \vec{g}^i)\Delta t^2 + \vec{v}_p^i(t-1)\Delta t + \vec{x}_p^i(t-1). \tag{21}$$

As we are concerned with relative position, we set the initial position to zero, and we also assume that the initial velocity is zero:

$$\vec{x}_p^i(0) = \vec{v}_p^i(0) = [0,0,0]. \tag{22}$$

The accelerometers' translation (its position relative to the initial frame) in terms of the rigid body rotation and translation is:

$$\vec{x}_p^i(t) = {}^tR^i\vec{x}_p^i(0) + \vec{x}_t^i. \tag{23}$$

Given this, we can compute the camera position at time t:

$$\vec{x}_t^i = {}^tR^i\vec{x}_p^i(0) - \vec{x}_p^i(t). \tag{24}$$

In Equation 20, it is necessary to subtract the value of gravity in the initial frame of the camera. We note, however, that the initial rotation of the camera relative to the world is unknown, as the gyroscopes only measure velocity. The accelerometers can be used to estimate the initial orientation if the camera initially has no external forces on it other than gravity. We have found this assumption

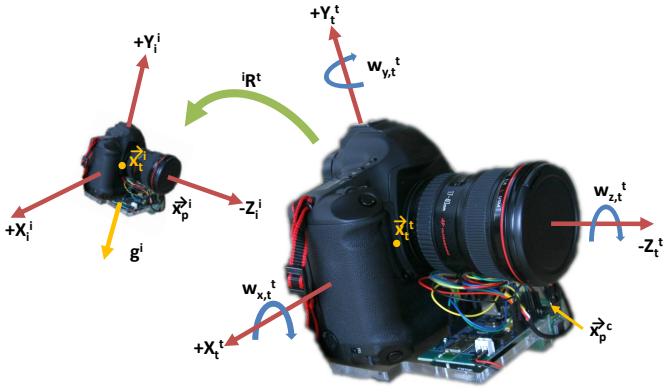

Figure 3: *The rigid-body dynamics of our camera setup.*

unreliable, so we instead make the assumption that the measured acceleration is normally distributed about the constant force of gravity. We have found this reliable when the camera motion is due to high-frequency camera-shake. Thus we set the direction of mean acceleration vector as the direction of gravity:

$$\vec{g}^i = mean(\vec{a}_p^i(t), [0...T])). \tag{25}$$

To summarize, the camera rotation and translation are recovered by integrating the measured acceleration and angular velocities that are rotated into the camera's initial coordinate frame. This gives us the relative rotation and translation over time, which is used to compute the spatially-varying PSF matrix in Equation 10. If the measurements are noise-free, this rotation and motion information is sufficient for deblurring; however, in practice, the sensor noise introduces significant errors. Furthermore, even if the camera motion is known perfectly, one still needs to know the scene depth, as discussed in Section 3. Thus additional steps are needed to deblur an image using the inertial data.

3.4 Drift Compensation and Deconvolution

It is well known that computing motion by integrating differential sensors can lead to drift in the computed result. This drift is due to the noise present in the sensor readings. The integration of a noisy signal leads to a temporally increasing deviation of the computed motion from the true motion.

We have measured the standard deviation of our gyroscope's noise to be $0.5deg/s$ and the accelerometer noise is $0.006m/s^2$, using samples from when the gyroscopes and accelerometers are held stationary (at zero angular velocity and constant acceleration, respectively) . In our experiments, there is significantly less drift in rotation, due to the need to perform only a single integration step on the gyroscope data. The necessity to integrate twice to get positional data from the accelerometers causes more drift.

To get a high-quality deblurring results, we must overcome the drift. We propose a novel aided blind deconvolution algorithm that computes the camera-motion, and in-turn the image blur function, that best matches the measured acceleration and angular velocity while maximizing the likelihood of the deblurred latent image according to a natural image prior.

Our deconvolution algorithms compensate for positional drift by assuming it is linear in time, which can be estimated if one knows the final end position of the camera. We assume the rotational drift is minimal. The final camera position is, of course, unknown; however, we know the drift is bounded and thus the correct final position should lie close to our estimate from the sensor data. Thus in contrast to a traditional blind-deconvolution algorithm that solves for each value of a kernel or PSF, our algorithm only has to solve for a few unknowns. We solve for these using an energy minimization framework that performs a search in a small local neighborhood around the initially computed end point.

In our experiments, we have found that the camera travels on the order of a couple millimeters during a long exposure (the longest we have tried is a 1/2 second). We note that a few millimeter translation in depth (z) has little effect on the image for lenses of common focal lengths, thus the drift in x and y is the only significant source of error. We set our optimization parameters to search for the optimal end point within a $1mm$ radius of the initially computed end point, subject to the constraints that the acceleration along that recovered path matches the measured accelerations best in the least-squares sense. The optimal end point is the one that results in a deconvolved image with the highest log-likelihood as measured by the hyper-Laplacian image prior (discussed in Section 3.1).

Specifically, let us define a function ρ that given a potential end point (u, v) computes the camera's translational path as that which best matches, in the least squares sense, the observed acceleration and terminates at (u, v):

$$\phi(\vec{a}^i, u, v) = \underset{\vec{x}^i}{\operatorname{argmin}} \sum_{t=0}^{T} (\frac{d^2\vec{x}_t^i}{dt^2} - \vec{a}_t^i)^2 + (\theta_{x,T}^i - u)^2 + (\theta_{y,T}^i - v)^2. \quad (26)$$

For notational convenience, let ρ define a function that forms the blur sampling matrix from the camera intrinsics, extrinsics, and scene depth as using the rigid-body dynamics and temporal integration processes discussed in Section 3.1 and 3.3:

$$A(d) = \rho(\vec{\theta}^i, \vec{x}^i, d, K) \quad (27)$$

The drift-compensated blur matrix and deconvolution equations are:

$$A(d, u, v) = \rho(\vec{\omega}^i, \phi(\vec{a}^i, u, v), d, K), \quad (28)$$

$$I = \underset{I,d,u,v}{\operatorname{argmin}} [||\vec{B} - A(d, u, v)\vec{I}||^2/\sigma^2 + \lambda||\nabla I||^{0.8}]. \quad (29)$$

We then search over the space of (u, v) to find the (u, v) that results in the image I that has the highest likelihood given the observation and image prior. We perform this energy minimization using the Nelder-Mead simplex method, and the spatially-varying deconvolution method discussed in Section 3.2 is used as the error function in the inner loop of the optimization. We perform the optimization on $1/10$ down-sampled versions (of our 21 MP images).

The results from our search process are shown as plots of the camera motion in Figure 5 and visually in Figure 7, when used to deblur images. The running time for this search method is about 5 minutes on a 0.75 MP image. The search only needs to be run once either for the entire image, or could be run on a subsection of the image if that is preferable. Once the drift is corrected for, the PSF is more accurate for the entire image.

Computing Scene Depth: Note that a spatially invariant scene depth is implicitly computed during the drift compensation process, as scaling the end point equally in the x and y dimensions is equivalent to scaling the depth value. Specifically, the optimization is over $u' = u/d$ and $v' = v/d$ and thus solves for a single depth value for the entire scene.

4 Deblurring System

In the previous section, we discussed how to remove camera motion blur by recovering the camera rotation and translation from accelerometers and gyroscopes. In this section, we describe our hardware for recording the accelerometer and gyroscope data and implementation related concerns and challenges.

4.1 Hardware Design

Since there are six unknowns per time-step, the minimal configuration of sensors is six. It is possible to recover rotation and translation using six accelerometers alone, by sensing each axis in pairs at three different points on a rigid-body; however, after experimenting with this method we found accelerometers alone to be too noisy for reliable computation of rotation. Thus our prototype hardware system, shown in Figure 1, is a minimal configuration consisting of a three-axis $\pm 1.5g$ MEMS accelerometer package and three single axis $\pm 150^\circ/s$ MEMS gyroscopes wired to an Arduino controller board with a Bluetooth radio. All parts are commodity, off-the-shelf components purchased online. Additionally, the SLR's hot-shoe, i.e., flash trigger signal, is wired to the Arduino board. The trigger signal from the SLR remains low for the entire length of the exposure and is high otherwise. The Arduino board is interrupt driven such that when the trigger signal from the SLR fires, the accelerometers and gyroscopes are polled at 200Hz during the exposure window. Each time the sensors are read, the values are sent over the Bluetooth serial port interface. Additionally, an internal high-resolution counter is read and the actual elapsed time between each reading of the sensors is reported.

Figure 4: *Ground Truth Blur Measurements: We attach a high-speed camera next to an SLR and capture high-speed video frames during the camera exposure. With additional wide-baseline shots, we perform 3D reconstruction using bundle adjustment. We show our high-speed camera attachment and a few images from the high-speed camera (we took about a hundred total for the process).*

The sensors and Arduino board are mounted to a laser-cut acrylic base that secures the board, the sensors, and a battery pack. The acrylic mount is tightly screwed into the camera tripod mount. The only other connection to the camera is the flash trigger cable.

Our hardware attachment has an optional feature used for calibration and validation experiments: mounting holes for a Point Grey high-speed camera. When the Arduino board is sampling the inertial sensors, it can also send a 100 Hz trigger to the high-speed camera. We use the high-speed data to help calibrate our sensors and to acquire data to get ground truth measurements for motion blur.

4.2 Calibration and Ground-truth Measurements

To accurately compute camera motion from inertial sensors, it is necessary to calibrate several aspects of our system. We need to accurately calibrate the sensor responses, as we have found them to deviate from the published response ranges. It is also necessary to know the position of the accelerometer relative to the camera's optical center. Lastly, we need to calibrate the camera intrinsics.

We calibrate these values in two stages. The first stage is to calibrate the sensors' responses. We do this by rotating the gyroscopes at a known constant angular velocity to recover the mapping from the 10-bit A/D output to degrees/s. We performed this measurement at several known angular velocities to confirm that the gyroscopes have a linear response. To calibrate the accelerometers, we held them stationary in six orientations with respect to gravity, $(\pm x, \pm y, \pm z)$, which allows us to map the A/D output to units of m/s^2.

To calibrate our setup and to measure ground-truth measurements for camera-shake, we developed a method to accurately recover a camera's position during an exposure using a method inspired by Ben-Ezra and Nayar [2004]. However, instead of tracking 2D motion, we track 6D motion. We attached a high-speed camera

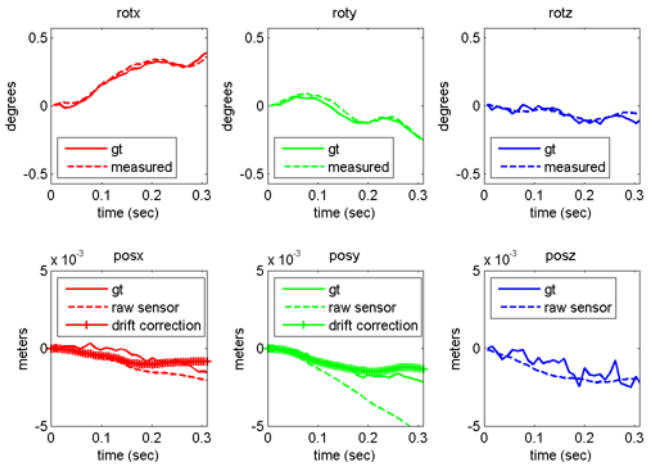

Figure 5: *Drift compensation: Angular and translational position of the camera versus time is shown for the image in the blue box in Figure 7. The "ground-truth" motion, computed using structure from motion, is shown as solid lines. The dashed lines are the motions from using the raw sensor data and the "+" marked lines are the result after drift compensation. The drift compensated results are much closer to the ground-truth result. Note the bottom right plot does not show a drift corrected line as we only compensate for x and y drift.*

(200 FPS PointGrey DragonFly Express) to our sensor platform, and our Arduino micro-controller code is set to trigger the high-speed camera at 100 FPS during the SLR's exposure window. In a lab setting, we created a scene with a significant amount of texture and took about 10 images with exposures ranging from $1/10$ to $1/2$ of a second. For each of these images, accelerometer and gyro data was recorded in addition to high-speed frames. We took the high-speed frames from these shots and acquired additional wide-baseline shots with the SLR and high-speed camera. Using all of this data, we created a 3D reconstruction of the scene using bundle adjustment (our process uses RANSAC for feature matching, computes sparse 3D structure from motion, and computes the camera focal length). Figure 4 shows our high-speed camera attachment and a few frames from the high speed cameras (we took about a hundred total with both cameras for the process).

This reconstruction process gives us a collection of sparse 3D points, camera rotations, and camera translations, with an unknown global transformation and a scale ambiguity between camera depth and scene depth. We resolve the scale ambiguity using a calibration grid of known size.

5 Results

We will now describe the results of our ground-truth camera-shake measurements and compare results using our deblurring method to the ground-truth measurements. We also compare our results to those of Shan et al. [2008] and Fergus et al. [2006], using the implementations the authors have available online. We also show results of our methods running on natural images acquired outside of a lab setup and compare these to results using previous work as well.

5.1 Lab Experiments and Camera-Shake Study

In Figure 6, we show visualizations of the ground-truth spatially-varying PSFs for an image from our lab setup. This image shows some interesting properties. There is a significant variation across the image plane. Also, the kernels displayed are for a fixed depth, thus all the spatial variance is due to rotation. To demonstrate the importance of accounting for spatial variance, on the bottom row of

Figure 6: *Visualization of ground-truth spatially-varying PSFs: For the blurry image on top, we show a sparsely sampled visualization of the blur kernels across the image plane. There is quite a significant variation across the image plane. To demonstrate the importance of accounting for spatially variance, in the bottom row we show a result where we have deconvolved using the PSF for the correct part of the image and a non-corresponding area.*

Figure 6, we show a result where we have deconvolved using the PSF for the correct part of the image and the PSF for a different, non-corresponding area. These results are quite interesting as they show that some of the common assumptions made in image deconvolution do not always hold. Most deconvolution work assumes spatially invariant kernels, which really only applies for camera motion under an orthographic model; however, with a typical imaging setup (we use a 40mm lens), the perspective effects are strong enough to induce a spatially-varying blur. We also note that there is often a roll component to the blur, something that is also not modeled by spatially invariant kernels. Lastly, we observe that translation, and thus depth dependent effects can be significant, which is interesting as it is often thought that most camera-shake blur is due to rotation. Please visit http://research.microsoft.com/en-us/um/redmond/groups/ivm/imudeblurring/ for examples.

In Figure 7, using the same scene as above, we show comparisons of deconvolution results using our method, ground-truth, and two others. The images shown of the lab calibration scene were taken at exposures from $1/3$ to $1/10$ of a second. For each image we show the input, the result of deconvolving with PSFs from the initial motion estimate and after performing drift correction, and compare these to a deconvolution using PSFs from the recovered ground-truth motions, and PSFs recovered using the methods of Shan et al. [2008] and Fergus et al. [2006]. For these latter two comparisons, we made a best effort to adjust the parameters to recover the best blur kernel possible. To make the comparison fair, all results were deblurred using exactly the same deconvolution method, that of Levin et al. [2007]. The results in Figure 7, show a wide variety of blurs, yet our method recovers an accurate kernel and provides deconvolution results that are very close to that of the ground-truth. In all cases, our results are better than those using Shan et al.'s and Fergus et al.'s methods.

5.2 Real Images

After calibrating our hardware system using the method discussed in Section 4.2, we took the camera outside of the lab, using a laptop with a Bluetooth adapter to capture the inertial sensor data. In Figure 8, we show several results where we have deblurred images using our method, Shan et al.'s method, and Fergus et al.'s method. The Shan et al. and Fergus et al. results were deconvolved using Levin et al.'s method [2007], and our results are deconvolved using our spatially-varying deconvolution method discussed in Section 3.1.

For all the images, our results show a clear improvement over the input blurry image. There is still some residually ringing that is unavoidable due to frequency loss during blurring. The Shan et al. results are of varying quality, and many show large ringing artifacts that are due to kernel misestimation and over-sharpening; the Fergus et al. results are generally blurrier than ours. All the images shown here were shot with $1/2$ to $1/10$ second exposures with a 40mm lens on a Canon 1Ds Mark III.

For additional results, visit http://research.microsoft.com/en-us/um/redmond/groups/ivm/imudeblurring/.

6 Discussion and Future Work

In this work, we presented an aided blind deconvolution algorithm that uses a hardware attachment in conjunction with a corresponding blurry input image and a natural image prior to compute per-pixel, spatially-varying blur and that deconvolves an image to produce a sharp result.

There are several benefits to our method over previous approaches: (1) our method is automatic and has no user-tuned parameters and (2) it uses inexpensive commodity hardware that could easily be built into a camera or produced as a mass-market attachment that is more compact than our prototype. We have shown advantages over purely image-based methods, which in some sense is not surprising—blind deconvolution is an inherently ill-posed problem, thus the extra information of inertial measurements should be helpful. There are many challenges to using this data properly, many of which we have addressed; however, our results also suggest several areas for future work.

The biggest limitation of our method is sensor accuracy and noise. Our method's performance will degrade under a few cases: (1) if the drift is large enough that the search space for our optimization process is too large, i.e., greater than a couple mm, (2) if our estimation of the initial camera rotation relative to gravity is incorrect or similarly, if the camera moves in a way that is not normally distributed about the gravity vector, (3) if there is significant depth variation in the scene and the camera undergoes significant translation, (4) if the camera is translating at some initial, constant velocity, and (5) if there is large image frequency information loss to blurring.

To handle (1), we are interested in using more sensors. The sensors are inexpensive and one could easily add sensors for redundancy and perform denoising by averaging either in the analog or digital domain. For (2) one could consider adding other sensors, such as a magnetometer, to get another measure of orientation – this is already a common approach in IMU based navigation systems. For (3) one could recover the depth in the scene by performing a "depth from motion blur" algorithm, similar to depth from defocus. We are pursuing this problem; however, it is important to note that varying scene depth does not always significantly affect the blur. For typical situations, we have found that depth is only needed for accurate deblurring of objects within a meter from the camera. Most often people take images where the scene is farther than this distance.

For (4) we assume the initial translation velocity is zero and our accelerometer gives us no measure of this. While we currently only consider the accelerometer data during an exposure, we actually record all the sensors' data before and after each exposure as well. Thus one way to address this issue is to try to identify a stationary period before the camera exposure and track from there to get a more accurate initial velocity estimate. (5) is an issue with all deblurring methods. Frequency loss will cause unavoidable artifacts during deconvolution that could appear as ringing, banding, or over-smoothing depending on the deconvolution method. It would be interesting to combine our hardware with the Raskar et al. [2006] flutter shutter hardware to reduce frequency loss during image capture.

Acknowledgements

We would like to thank the anonymous SIGGRAPH reviewers. We also thank Mike Sinclair and Turner Whitted for their help in the hardware lab.

References

BASCLE, B., BLAKE, A., AND ZISSERMAN, A. 1996. Motion deblurring and super-resolution from an image sequence. In *ECCV '96: Proceedings of the 4th European Conference on Computer Vision-Volume II*, Springer-Verlag, London, UK, 573–582.

BEN-EZRA, M., AND NAYAR, S. K. 2004. Motion-based motion deblurring. *IEEE Trans. Pattern Anal. Mach. Intell. 26*, 6, 689–698.

CANON, L. G. 1993. *EF LENS WORK III, The Eyes of EOS*. Canon Inc.

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

JOSHI, N., SZELISKI, R., AND KRIEGMAN, D. J. 2008. Psf estimation using sharp edge prediction. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 1–8.

KUNDUR, D., AND HATZINAKOS, D. 1996. Blind image deconvolution. *Signal Processing Magazine, IEEE 13*, 3, 43–64.

LEVIN, A., FERGUS, R., DURAND, F., AND FREEMAN, W. T. 2007. Image and depth from a conventional camera with a coded aperture. *ACM Trans. Graph. 26* (July), Article 70.

LEVIN, A., SAND, P., CHO, T. S., DURAND, F., AND FREEMAN, W. T. 2008. Motion-invariant photography. *ACM Trans. Graph. 27* (August), 71:1–71:9.

LEVIN, A., WEISS, Y., DURAND, F., AND FREEMAN, W. 2009. Understanding and evaluating blind deconvolution algorithms. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, IEEE Computer Society, 1964–1971.

PARK, S.-Y., PARK, E.-S., AND KIM, H.-I. 2008. Image deblurring using vibration information from 3-axis accelerometer. *Journal of the Institute of Electronics Engineers of Korea. SC, System and control 45*, 3, 1–11.

RASKAR, R., AGRAWAL, A., AND TUMBLIN, J. 2006. Coded exposure photography: motion deblurring using fluttered shutter. *ACM Trans. Graph. 25* (July), 795–804.

RICHARDSON, W. H. 1972. Bayesian-based iterative method of image restoration. *Journal of the Optical Society of America (1917-1983) 62*, 55–59.

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

STEWART, C. V. 1999. Robust parameter estimation in computer vision. *SIAM Reviews 41*, 3 (September), 513–537.

TAI, Y.-W., DU, H., BROWN, M. S., AND LIN, S. 2008. Image/video deblurring using a hybrid camera. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 1–8.

YUAN, L., SUN, J., QUAN, L., AND SHUM, H.-Y. 2007. Image deblurring with blurred/noisy image pairs. *ACM Trans. Graph. 26* (July), Article 1.

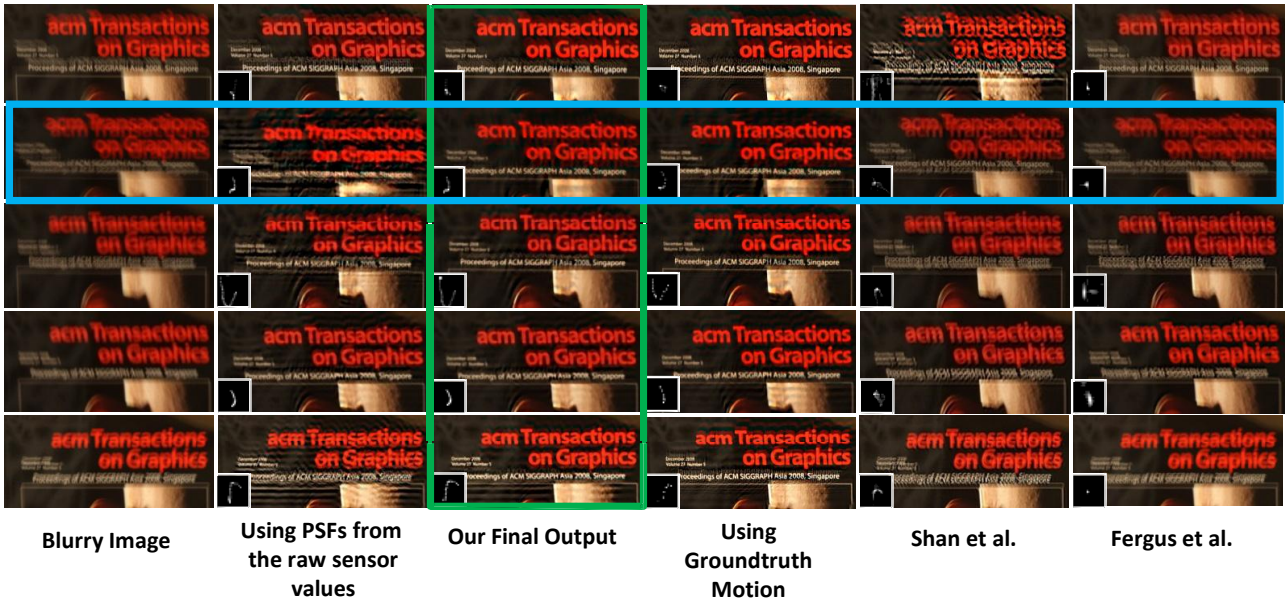

Figure 7: *Deblurring results and comparisons: Here we show deblurring results for a cropped portion of the scene shown in Figure 6. These sections are cropped from an 11 mega-pixel image. The results in the green box are the final output of our method, where the sensor data plus our drift compensation method are used to compute the camera motion blur. Subtle differences in the PSF before and after drift compensation can have a big result on the quality of the deconvolution.*

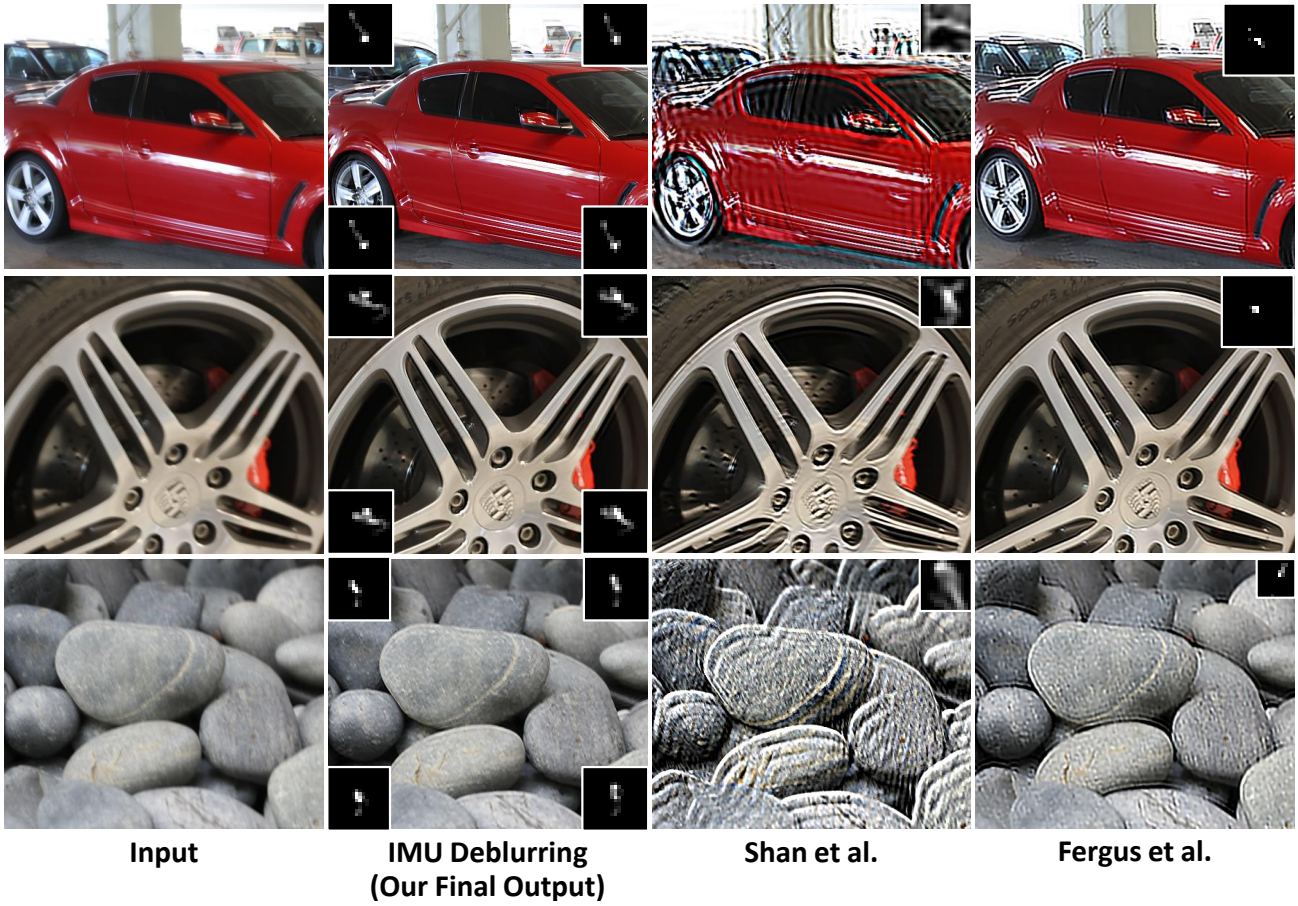

Figure 8: *Natural images deblurred with our setup: For all the images our results show a clear improvement over the input blurry image. The blur kernel at each corner of the image is shown at $2\times$ size. There is still some residual ringing that is unavoidable due to frequency loss during blurring. The Shan et al. results are of varying quality, and many show large ringing artifacts that are due to kernel misestimation and over-sharpening; the Fergus et al. results are generally blurry than our results. With the stones image, the intentionally defocused background stones are not sharpened in our result.*