Motion Aware Motion Invariance

Scott McCloskey, Kelly Muldoon, and Sharath Venkatesha
Honeywell ACS Labs
1985 Douglas Drive North #112A, Golden Valley, MN 55422
{scott.mccloskey, kelly.muldoon, sharath.venkatesha}@honeywell.com

Abstract

We address motion de-blurring using a computational camera that captures an image while the stabilizing optical element moves in a modified Canon IS lens. Our work builds on that of Levin et al. *[11], who introduce parabolic motion as a means of achieving invariance to unknown subject velocity in an a priori known direction. While the previous work addresses a specific scenario - exact knowledge of motion orientation and a uniform, symmetric prior on its magnitude - we generalize this to address scenarios where the motion of objects in the scene or the camera itself are known to various extents. We describe a motion invariant camera based on an off-the-shelf lens, and show how its motion and position sensors can be used to inform both the image capture and de-blurring. We demonstrate that our changes to motion invariance improve the quality of captured images in the case of both object and camera motion.*

1. Introduction

Motion blur is a nuisance that reduces both the aesthetic appeal and the utility of images captured when objects in the scene are not stationary. Likewise, motion blur can arise when the camera itself moves during exposure. While motion de-blurring is a long-standing area of research, the field has broadened in recent years to include the capture and processing of moving scenes by computational cameras. Much of the attention has focused on camera designs which ensure that texture information about the scene is encoded in a photographic data structure *from which a sharply-focused image can be derived by de-blurring*. Because blur estimation - from computational or traditional cameras - is ill-posed, motion invariance [11] was developed to invertibly capture scene texture *without the need for blur estimation*. This is achieved by moving some part of the camera along a line during image capture, so that all objects in the scene (including stationary ones) are encoded with approximately the same motion Point Spread Function (PSF). The motion invariant image can thus be de-convolved with a PSF determined by *a priori* calibration, obviating the need for blur estimation *as long as object motion is parallel to the camera motion*. Thus, motion invariance is designed for cases where motion orientation is known but magnitude is unknown.

The constant acceleration used in motion invariance is selected to allocate the bandwidth optimally for the unknown motion magnitude, and amounts to a uniform prior on the expected distribution of object speed. While this is broadly applicable, the approximate nature of the invariance to object motion means that this uniform prior will give suboptimal performance when the actual distribution of object speed is non-uniform. Similarly, the uniform prior does not incorporate readily available data from motion and position sensors already in the lens.

In order to address these shortcomings, we introduce *motion aware* motion invariance as a means to improve image quality via alternative priors on subject motion, and by incorporating data from the camera's sensors. At a concrete level, we demonstrate (1) a noise-optimal capture approach for scenes with asymmetric motion priors, (2) a per-image PSF calibration based on actual lens motion during exposure, and (3) cancellation of camera body motion in order to achieve consistent invariance to object motion. Each of these is tightly coupled with our implementation of motion invariance using optical components present in Digital Single Lens Reflex (DSLR) lenses with optical Image Stabilization (IS). Throughout, we avoid image-based blur estimation so that the key benefit of motion invariance - not needing a pre-exposure velocity estimate - still holds.

2. Previous Work

Our work builds on Levin *et al*. [11], who introduced motion invariance, and demonstrated the optimality of parabolic motion in providing the desired invariance. They later extend the motion invariance concept to cases where the motion direction is not known [3], by capturing two images with orthogonal motion, which they implemented by shifting the sensor. More recently, other implementa-

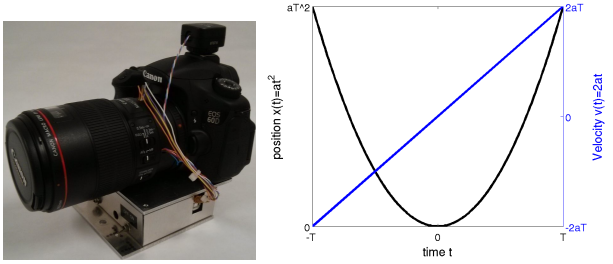

Figure 1. (Left) Our motion invariant camera, based on a standard Canon 60D SLR with a modified 100mm Macro IS lens. (Right) Motion invariant capture uses parabolic motion in position versus time (black curve, left axis) to capture an image where all velocities in a range are stabilized for the same period of time due to the linearity of velocity as a function of time (blue line, right axis).

tions of motion invariance have been described in the literature. Sonoda *et al*. [18] describe an implementation of motion invariance using a Liquid Crystal on Silicon (LCoS) element, though that implementation involves about three stops of light loss due to the combined effects of polarization, fill factor, reflectivity, and the reduction of the reflective region needed to implement the translating aperture. We have previously [15] described a prototype implementation of motion invariance using the image stabilization hardware in a Canon DSLR lens. Relative to the prior work, this manuscript introduces the concept of motion aware motion invariance, where prior knowledge of motion - including camera motion estimated from sensors embedded in the stabilized lens - is used to achieve improved invariance to subject motion. In this sense, or work is related to Joshi *et al*. [7], who add inertial sensors to a DSLR body to constrain blur estimation for traditionally-captured images. We do something similar for motion invariant images, and we also use the data from these sensors to change the image capture process itself.

Besides motion invariance, several computational photographic techniques have been developed to address motion blur. Notably, coded exposure [17] addresses the lack of motion blur invertibility in traditional imaging, but its lack of invariance requires pre-exposure motion estimation in order to guarantee invertibility [12]. Coded exposure blur estimation has been developed using either edge priors [2] or natural image statistics [14]. Both coded exposure and motion invariance have analogous approaches for handling optical blur, as well. Coded aperture photography [10, 19] selectively occludes light paths to the sensor, and has been used for extended depth of field imaging. Similar to motion invariance, flexible depth of field photography [16] captures a coded image using lens motion, though the lens moves along the optical axis and achieves invariant defocus blur over an extended depth of field. These and other computational cameras are described by Zhou and Nayar [22].

Despite the associated invertibility problems, there is a large amount of research addressing both blur estimation and de-blurring of images from traditional cameras. Recently, work in this area has addressed spatially variant blur arising from camera rotations [20, 4]. Interestingly, experiments with a dataset of hand-held camera motions [8] have shown that these methods don't always out-perform those based on a shift-invariant model of motion blur. However, [8]'s evaluation results on blind de-convolution do not necessarily reflect performance on motion invariant deblurring, where the invariant PSF is known ahead of time. We use the method of Krishnan and Fergus [9] for the non-blind deconvolution in our experiments, and note that it also performs well in the blind deconvolution evaluation.

3. Review of Motion Invariance

Motion invariance can be implemented by translating any of the camera body, sensor, or lens with constant acceleration. We use optical stabilization hardware to implement motion invariance by lens motion, as suggested by Levin *et al*. in [11, 3]. In the 1D case, the lens moves along a line in the direction of expected motion; without loss of generality, we will discuss horizontal lens motion with initial rightward velocity. At the beginning of the image's exposure, the lens translates right with a given velocity, and constant (negative) acceleration a is applied. During an exposure duration of $2T$ milliseconds, with the time variable $t \in [-T, T]$ for mathematical convenience, the acceleration causes the lens to come to a stop at $t = 0$ and, at the end of the exposure period ($t = T$), the lens has returned to its initial position with the same velocity magnitude (but in the opposite direction) as in the beginning. Though the motion of the lens is linear, this pattern is referred to as *parabolic motion* because the horizontal position x is a parabolic function of time t,

$$x(t) = at^2. \tag{1}$$

Figure 1 shows parabolic motion and lens velocity (the derivative of x), illustrating the range of object velocities which are stabilized during exposure.

As in other work, blur in a motion invariant image is modeled as the convolution of a latent image I with a PSF B, plus additive noise η, giving the blurred image

$$I_b = I * B + \eta. \tag{2}$$

4. Motion Invariance via Stabilization

Our motion invariant camera (Fig. 1) is made of a stock Canon 60D DSLR body with a modified Canon EF 100mm f/2.8L Macro IS lens, the key components of which are sensors that record camera motion and a stabilizing lens element that can be moved. The standard mode of operation

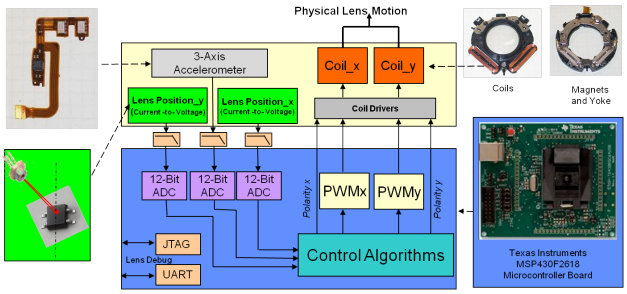

Figure 2. System block diagram of our motion invariant lens. We add a microcontroller (blue box) to induce motion using the hardware already contained in the Canon IS lens (yellow box). Illustrations of the hardware elements (left and right columns) are connected to the corresponding block by dashed arrows.

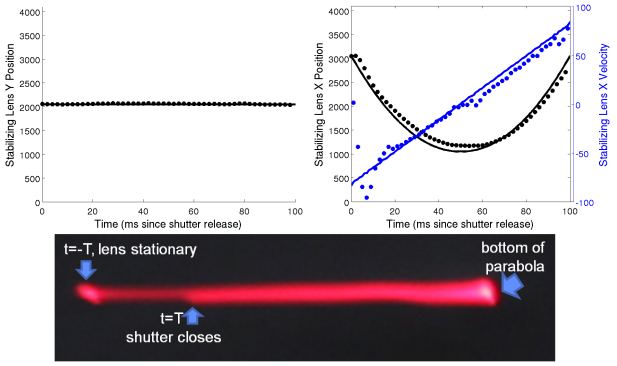

Figure 3. Desired (solid line) versus actual (points) motion. Left plot shows lens Y position for a horizontal motion. Right plot shows lens X position (black, left axis) and velocity (blue, right axis). Positions are given in 12 bit ADC samples. Below we show the corresponding image of a stationary LED light source.

for image stabilization is a closed control loop where motion detected by sensors within the lens induces a compensating motion of the stabilizing element. In order to drive the lens to parabolic motion, we break the control loop and decouple the stabilizing element from the motion sensor. We add a Texas Instruments MSP430F2618 microprocessor to run control loops executing the desired parabolic motion, which is triggered by the camera via the hot shoe signal.

Fig. 2 shows a block diagram of the lens control implementation we use in our prototype, with the yellow box encapsulating hardware within the lens and the purple box encapsulating our added microcontroller. The IS lens position is determined by 2 optoelectronic position sensors shown in the green boxes, with high resolution position sensing. Two PSDs are used in the system for sensing position in the X and Y positions. The voltage output of the sensors are tapped off the system, low-pass filtered and then are sampled at 1KHz by internal 12-bit ADCs on the microcontroller. The ADC samples are the inputs to a control algorithm which drives two Pulse Width Modulators, which are fed into the coil drivers providing the physical motion of the IS lens. Polarity signals associated with X and Y position coil drivers control motion direction (left/right and up/down).

Unlike the implementation of McCloskey *et al.* [15], where a host computer was needed to provide motion commands during exposure, the current prototype stores a sequence of desired lens positions on the controller's flash memory so that the camera can operate untethered. We have also replaced the camera body with a stock Canon DSLR, which provides standard functionality (auto-focus, -exposure, etc.) that was lacking in McCloskey's prototype. We also capture motion data from the lens's accelerometers and gyros, the uses of which are described in Section 7.

A classic PID^2 controller is applied to both X and Y positions in the system. Desired lens positions from flash memory are fed to the X and Y controllers and the PID^2 loop runs continuously as new destinations are introduced. Because we use a 12 bit ADC, the lens has 4096-by-4096 discrete positions over a range of approximately 3.5mm-by-3.5mm. We find that movement between points in the periphery was not reliable, so we restrict our motion to the central 2048-by-2048 region of ADC positions. In the 8MP image format that we use in our experiments, lens motion of 9.4 ADC samples corresponds to 1 pixel, meaning that the motion invariant PSF is about 218 pixels.

When the camera's shutter release is depressed half way, the mechanical lock on the stabilizing lens is released and the controller drives it to the first (x, y) position in the sequence; the lens also focuses normally. When the camera's shutter release is fully depressed, the shutter opens, the controller receives a signal from the hot shoe, and then visits the sequence of (x, y) points stored in memory.

Figure 3 illustrates the performance of our motion control, showing the desired versus actual motion of the lens along the X and Y axes. The left plot illustrates that there is a very small amount of unintended motion along the Y axis. The right plot shows the intended parabolic motion and velocity along the X axis, along with the actual motion and velocity. Due to a delay between the shutter opening and the front curtain hot shoe signal, the lens rests at its starting position for about 2ms before motion starts. This results in a high velocity error for the first 4ms, after which the actual velocity matches the intended value quite well. Also, the delay prevents the the lens from returning to its starting position by the end of the exposure, as is illustrated by the image of a stationary LED light source. This image reflects the motion PSF, which has two peaks: one arising from the intentional slowdown at the bottom of the parabola, and the second from the delay at the beginning of exposure. We illustrate the impact of this second peak in the next section.

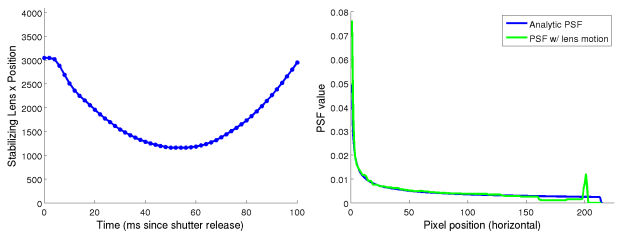

Figure 4. Using the observed stabilizing lens positions to generate the PSF for de-blurring. Left plot shows observed X positions of the lens over a 100ms exposure (Y values have minimal motion, and are ignored). The right figure plots the analytic PSF (blue curve), along with the PSF using observed lens positions (green).

5. Motion Aware De-blurring

As with any mechanical system, our control system for lens motion will not perfectly reproduce the intended parabola. Moreover, the motion will vary from one exposure to the next, due to fine-scale differences in, for example, the amount of time between the shutter's half and full depression (during which time the lens goes to the designated start position). While these motion imperfections are unavoidable, they need not reduce the quality of the resulting image. Because our hardware already captures lens position during exposure for the purposes of motion control, it is a natural to use this data in de-convolution.

During exposure, the microcontroller saves the actual (x, y) lens position to flash memory. To generate the real camera PSF, we fit a linear spline to the recorded x and y positions, as shown in Figure 4. We then interpolate the time value at intersections of the lens position with pixel boundaries, i.e. parallel lines spaced every 9.4 ADC values, and differentiate in order to determine the amount of time light from a moving point fell on that pixel. The time differentials are normalized by the total exposure time in order to produce a PSF with unit power. Because the initial position of the lens with respect to the pixel boundaries is unknown, and in order to reduce noise inherent in differentiating our position values, we repeat this process for several different initial sub-pixel positions and average them to get the motion-aware PSF (Fig. 4, right). This PSF accounts for the delay in motion onset, with both a second peak (around $x = 200$) and a reduced values in $170 \leq x \leq 195$ due to the lens not completely returning to the start position.

Figure 5 (best viewed electronically) shows the same motion invariant image de-blurred using PSFs with/without accounting for the real motion. From the insets, we see that the PSF determined from the real lens motion (bottom row) has a reduced echo on the stationary blue car than the analytic PSF (top row), which does not account for the hot shoe delay.

Figure 5. Comparing the performance of the PSFs from Fig. 4. Top row: de-blurred using the analytic PSF. Bottom row: de-blurred using actual lens motion, which reduces ghosting artifacts. Best viewed electronically.

6. Asymmetric Motion Priors

While estimating the motion of individual objects is avoided by motion invariance, the use of prior information about the range of motion in the scene is fundamental to the approach. As described in [11], the expected range of object velocities determines the value of the acceleration parameter a. Because the range of velocity invariance is $[-2at, 2at]$, and considering that higher values of a lead to wider blur PSFs (i.e., more noise after reconstruction), getting the best result from motion invariance requires an accurate motion prior. However, choosing a only allows us to choose a motion prior which is both symmetric and uniform. In many cases, such as a traffic camera overlooking several adjacent highway lanes or a one-way street, the correct motion prior is asymmetric.

Consider a scene where our velocity prior is uniform over a range $[0, S]$. Though it will be sub-optimal in the sense of the bandwidth budgeting analysis presented in [11], the baseline approach is to use a full parabola over an exposure time of $2T$ and acceleration a such that $2aT = S$. Relative to this baseline, the obvious way to handle asymmetric motion priors is to move the lens in a half parabola which only stabilizes objects moving in the expected direction. Because the range of the stabilizing lens's motion is limited by the size of the cavity in which it's located, executing a half parabola with acceleration a only takes Tms,

effectively reducing exposure by one stop. In these situations, where we assume the lens aperture can't be widened (e.g., because it would reduce depth of field) to keep exposure constant, there are two options:

1. Increasing gain (ISO) and using exposure time T, in order to use the same value of a and preserve the extent of motion invariance. The disadvantage of this approach is that the reduction in the number of photons collected will reduce the SNR of the captured image.

2. Alternatively, we can reduce the acceleration a, use exposure time $2T$ for a high photon count. This means that the range of lens velocities is reduced, implying a reduction in the extent of motion invariance.

While it is clear that option 2 will produce a better image for stationary objects, it is not obvious which of these approaches will produce the better quality image for non-zero object velocities. To quantify this tradeoff we have modeled and simulated the two options. For option 1, we need a model for the change of the variance of noise η as a function of changing sensor gain. We quantify gain in terms of the camera's ISO setting, and adapt the noise model used for HDR capture optimization in [5]. The change in noise variance as a function of changing ISO from G_1 to G_2 is

$$Var(\eta_1) - Var(\eta_2) = \frac{\Phi 2T + \sigma_{read}^2}{U^2}(G_1^2 - G_2^2), \quad (3)$$

where $\Phi 2T$ is the signal dependant term, σ_{read} is due to read noise, and U is a camera dependant constant. Since we are interested in comparing two strategies for the same camera, capturing the same scene with a fixed exposure time, we treat the fraction as a constant k_{ISO} such that

$$Var(\eta_1) - Var(\eta_2) = k_{ISO}(G_1^2 - G_2^2). \quad (4)$$

We captured several images of a stationary scene at different ISO settings in order to calibrate the camera-dependant value of k_{ISO}[1]. Using this value and equation 4, we simulate blurring (eq. 2) and compute RMSE after deconvolution. We do this for 10 photographic images from [6], and compute the average RMSE for each several stops of ISO increase, for several values of $s \in [0, aT]$, and generate the performance curves for option 1 shown in Figure 6. What this shows is that de-convolution performance is consistent for different values of s, but that the increase in ISO leads to decreasing performance (higher RMSE) in low-light settings.

Evaluating option 2 (increasing the exposure time while keeping ISO constant) requires only that we simulate the motion invariant PSF for a range of velocities against a lens acceleration a which changes as a function of light loss.

[1]For our Canon 60D, $k_{ISO} = 1.1233e-5$.

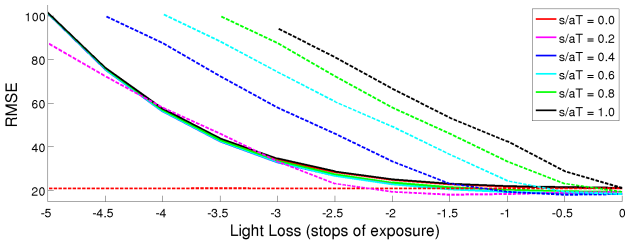

Figure 6. Comparing the reconstruction performance of low-light motion invariance options over a range of subject velocities. Solid lines (mostly overlapping) show the performance of increasing ISO while keeping exposure time constant (option 1). Dashed lines show the performance of increasing the exposure time while keeping ISO constant (option 2).

With constant lens displacement D, a changes in response to increasing exposure time from $2T_1$ to $2T_2$ as

$$a_1 T_1^2 = D = a_2 T_2^2. \quad (5)$$

So, when available light is reduced by one stop, T doubles and a is reduced by a factor of 4. The simulated impact of this is shown alongside option 1 in Figure 6.

Comparing the RMSE impacts of the two alternatives, we see that option 2 (increasing exposure time) provides consistent performance for stationary objects. However, the performance of option 2 decreases quickly with light for non-stationary objects, performing better than option 1 for $s < \frac{aT}{5}$ and worse for larger values. Qualitatively, option 1's RMSE is due to amplification of the noise component which does not depend on s, whereas option 2's RMSE is due to increasingly strong artifacts with increasing s. Due to this, and consistent with an intuition that moving parts of the scene are more interesting than stationary ones, we adopt option 1 and increase ISO for half parabola capture.

The real camera images in Figure 7 demonstrate that the half parabola motion with increased ISO provides a higher-quality reconstruction than the full parabola for objects moving in the expected direction. The tradeoff, of course, is that the half parabola can not stabilize objects that disobey the prior; if the motion is in the wrong direction, it will not be stabilized by the lens motion and the PSF will have no peak. As a result, the reconstruction will be of low quality, comparable to a traditional camera's image.

7. Velocity Aware Motion Invariant Capture

Though it does not handle the complex in- and out-of plane rotations addressed by recent image de-blurring algorithms, motion invariance can still hold for certain camera motions. Namely, a linear, constant-velocity motion of the camera in a direction parallel to the sensor plane. Instances of such blur are less common, but include cases like taking

Figure 7. Comparing the performance of a half parabola (left column) with the full parabolic motion invariance (right column). When the motion prior is asymmetric, and a half parabola is used to capture rightward motion, the reconstruction has higher contrast and less noise. The first row shows a scene with rightward motion, and the second row shows a closeup of the train. Reconstruction quality is higher from the half parabola, though both images have reconstruction artifacts due to specular reflections off of the metal parts of the wheel. When the motion does not obey the asymmetric prior distribution, the half parabola cannot give a good reconstruction (third row, left); a traditional image capture with the same object velocity and exposure time is shown (third row, right).

image from a moving car. In this section, we first investigate the implications of such camera motion on the range of invariance to object motion. Then, we describe how the accelerometers and gyros included in our stabilizing lens can be used to estimate camera velocity immediately before the start of exposure. Finally, we demonstrate that the addition of an offset velocity in the parabolic motion can be used to compensate for a limited range of camera velocity.

7.1. The Effect of Camera Motion on Invariance

Suppose that we have a camera translating laterally with constant velocity v_c during the exposure of a motion invariant image. In past work, the motion of the stabilizing lens would be executed as usual, giving a parabola and symmetric velocity range about zero *relative to the camera's reference frame*. Considering the camera velocity, the motion of the stabilizing lens element in the world reference frame becomes

$$x(t) = at^2 + v_c t, \tag{6}$$

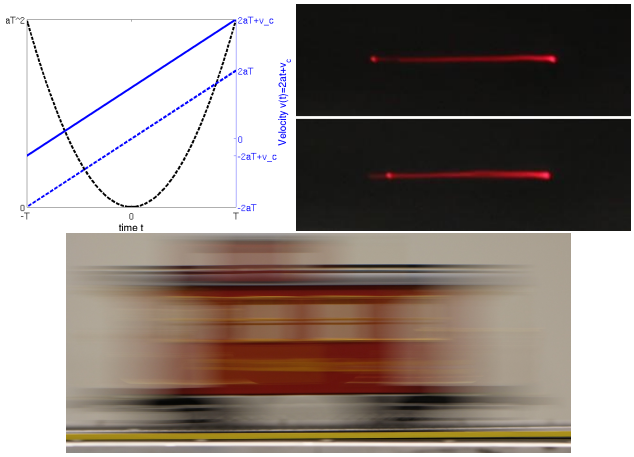

Figure 8. The effect of a non-zero camera velocity on motion invariance. Within the camera's reference frame, parabolic motion and a symmetric range of velocities are still achieved (dashed lines). Within the world reference frame, however, the camera's velocity offsets the lens velocity and changes the range of motion invariance from $[-2aT, 2aT]$ to $[-2aT + v_c, 2aT + v_c]$ (solid line). Right images show the appearance of a stationary red dot in the world through a stationary camera (top) and a moving one (bottom), both executing the same parabolic lens motion. The bottom image shows a motion invariant image which fails to stop the motion of a moving train due to non-zero camera velocity. No useful de-blurred result can be recovered from this image.

and the lens velocity becomes

$$v(t) = 2at + v_c. \tag{7}$$

This shifts the range of velocity invariance from $[-2aT, 2aT]$ to $[-2aT + v_c, 2aT + v_c]$, as illustrated in Figure 8. That figure also illustrates that the camera's velocity causes a stationary object in the world to have the same PSF as an object with velocity v_c observed from a stationary motion invariant camera. Presuming that the lens acceleration a was chosen to provide a symmetric invariance range of $[-2aT, 2aT]$, the presence of camera motion will be problematic. Objects with velocities in the range $[-2aT, -2aT + v_c)$, for instance, will no longer be stabilized and will not have reasonable quality in the de-blurred image. The image in Figure 8 illustrates such a case, where the shift in the range of invariance prevents the stabilization of a moving train. No reasonable reconstruction can be produced from this image. In order to avoid this unintended consequence of camera motion, we propose to measure the camera velocity and compensate for it by modifying the lens's motion.

7.2. Velocity Estimation from Lens Sensors

In order to compensate for this effect of camera motion on the range of object motion invariance, it is first necessary

to estimate the camera's velocity. While the accelerometers and gyros included in our stabilizing lens are generally used for relative measurements during exposure, these same sensors can be used to estimate absolute camera velocity. Our microcontroller captures the output of the 3 axis accelerometer, and samples data $\overrightarrow{a} = (A_x, A_y, A_z)$ after a low-pass filter is applied. At a high level, velocity is estimated by integrating the accelerometer's output, with two complications. First, integrated velocities from the accelerometer will tend to drift over long periods of time; in order to avoid this, we add a reset button which zeros the integration and do so when the camera is at rest. Second, while the accelerometer's output measures physical acceleration, it has an additive component of $9.8m/s^2$ in the direction of gravity. The separation of these two components is ill-posed when the accelerometer's three measurements are the only data available, and is made well-posed when the gyro outputs are used in estimation.

For the limited purpose of demonstrating motion aware motion invariance, we avoid the engineering effort needed to solve the source separation problem. We average the accelerometer outputs over a three second interval when the camera is at rest, and to use this as the estimate of gravity

$$\overrightarrow{\hat{g}} = (\overline{A_x}, \overline{A_y}, \overline{A_z}). \tag{8}$$

We then carry out experiments where the camera pose is fixed (i.e., we eliminate rotations) by mounting the camera on a robot arm. This allows us to estimate velocity by

$$\overrightarrow{v_c} = \int_t (\overrightarrow{a} - \overrightarrow{\hat{g}})dt. \tag{9}$$

7.3. Compensating for Camera Motion

Having estimated the camera velocity v_c from the lens's accelerometers, the intuitive modification of the lens motion is to add a compensating $-v_c$ velocity term to the lens motion within the camera coordinate frame,

$$v_{lens}(t) = 2at - v_c, \tag{10}$$

and to integrate from here to get the lens position (within the camera's coordinate frame)

$$x_{lens}(t) = at^2 - v_c t + x_0. \tag{11}$$

One complication which arises is that the starting position of the lens $x_{lens}(-T)$ can not depend on the velocity estimate because, when the photographer presses the shutter release halfway, the controller must move the lens to the starting position *and the camera velocity may continue to change.* Our solution is to fix the starting lens position and to make x_0 a function of v_c, so that $x_{lens}(-T) = x_{start}$ for all values of v_c, by

$$x_0(v_c) = x_{start} - aT^2 + v_c T. \tag{12}$$

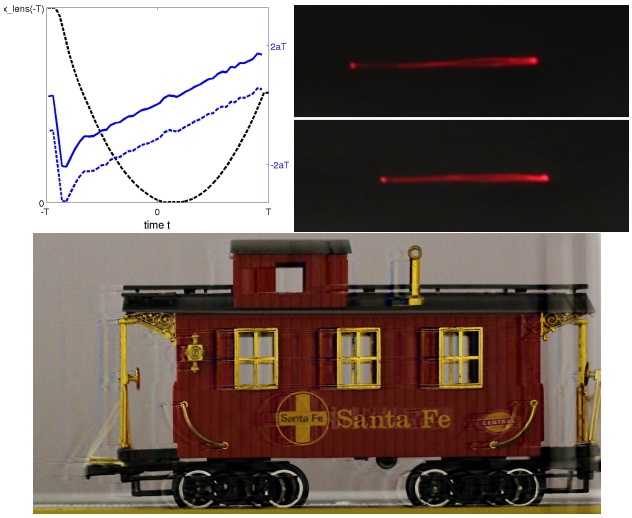

Figure 9. Compensating for non-zero camera velocity in motion invariance. Within the camera's reference frame, compensated parabolic motion and a range of velocities are asymmetric (dashed lines). Within the world reference frame, however, the camera's velocity offsets the lens velocity and centers the range of stabilized velocities at 0. Right images show the appearance of a stationary red dot in the world through a stationary camera (top) and a moving one (bottom). The bottom row shows the de-blurred image of a train moving in the opposite direction of the camera, which can only be stabilized due to the velocity compensation performed by our motion aware motion invariant camera.

In practice, we pre-compute the values of x_0 and x_{lens} for several values of v_c, store them in the controller's flash memory, and use the estimate of v_c at the time of the shutter release to select which to execute. While the accelerometer can measure fairly large velocities, our ability to compensate for high velocities is limited by the amount of displacement D the lens can achieve. Even if we allow the lens to travel outside of the 2048-by-2048 ADC area at the center of the field of motion, the maximum camera velocity that we can cancel is

$$v_c \leq \frac{0.04m}{T}. \tag{13}$$

Here we have another situation where minimizing exposure time produces a higher benefit, and where increasing ISO to keep the invariant range is preferred to reducing a.

Figure 9 demonstrates the compensated motion trajectory, lens velocity in the camera reference, and lens velocity in the world reference. In this case, the estimated velocity is about 0.08m/s. Despite not being very fast, this level of camera motion prevented the symmetric parabola from achieving stabilization on an object moving in the opposite direction. But stabilization is achieved during the exposure with our compensated trajectory, allowing for the reconstruction of a sharp image of the same object.

8. Limitations and Assumptions

With the exception of the last section - which applies to a restricted class of camera motion - our work is intended for stationary cameras and we do not compensate for camera rotation during exposure. While Cho *et al*. [3] have demonstrated how this can be extended to arbitrary motion directions, a key assumption of our motion aware motion invariance is that the orientation of subject motion is known a priori. In practice we do not find this particularly limiting to the stationary camera scenario, in that the motion of interesting objects (people, cars, animals, etc.) is generally due to locomotion along a ground plane.

9. Conclusion and Future Work

In the previous sections, we have described how motion invariance can be extended to incorporate motion priors other than the uniform symmetric prior used previously, and how actual motion - of the lens or the whole camera body - can be compensated for during image capture or deblurring. Our contributions have been:

1. Developing optimal capture for asymmetric motion priors, and demonstrating its improvement with half parabola image captured using our camera.
2. Incorporating actual lens motion to produce a higher quality reconstruction of images from our camera.
3. Analyzing the impact of camera motion on the range of object motion invariance. Using the accelerometers and gyros already included in the lens, we estimate camera motion and modify the lens motion profile in order to prevent a change in the range of invariance.

Though we have demonstrated these in isolation in order to clearly illustrate their improvements relative to baseline performance, each of these can be combined on a per-image basis as necessary.

Our research has been motivated by the use of motion invariance for recognition and matching tasks. To compliment the image quality comparisons of motion invariance and coded exposure [3, 1], we plan a future comparison of the methods based on their performance in recognition systems. In particular, we plan to compare the performance of motion invariant iris and barcode capture to our previous work with coded exposure [13, 21]. The main question is how remaining ghosting artifacts, which arise from the approximate nature of motion invariance, impact recognition performance.

More generally, we believe that the lens's motion sensor can be used to capture images with a consistent motion invariant range despite camera motion arising from camera shake. Using the motion data captured by Köhler *et al*. [8], our future work will include development of real-time control algorithms to cancel hand-held camera shake while capturing a motion invariant image.

Acknowledgement

This material is based upon work supported by the U.S. Army RDECOM under Contract Number W911NF-11-C-0253. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. Government. All brand names and trademarks used herein are for descriptive purposes only and are the property of their respective owners.

References

[1] A. Agrawal and R. Raskar. Optimal single image capture for motion deblurring. In *Computer Vision and Pattern Recognition*, pages 2560–2567, 2009.

[2] A. Agrawal and Y. Xu. Coded exposure deblurring: Optimized codes for PSF estimation and invertibility. In *Computer Vision and Pattern Recognition*, 2009.

[3] T. S. Cho, A. Levin, F. Durand, and W. T. Freeman. Motion blur removal with orthogonal parabolic exposures. In *Int'l Conf. on Computational Photography*, 2010.

[4] A. Gupta, N. Joshi, L. Zitnick, M. Cohen, and B. Curless. Single image deblurring using motion density functions. In *European Conf. on Computer Vision*, 2010.

[5] S. W. Hasinoff, F. Durand, and W. T. Freeman. Noise-optimal capture for high dynamic range photography. In *Computer Vision and Pattern Recognition*, pages 553–560, 2010.

[6] J. Hays and A. A. Efros. im2gps: estimating geographic information from a single image. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008.

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

[8] 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 Conf. on Computer Vision*, pages 27–40, 2012.

[9] D. Krishnan and R. Fergus. Fast image deconvolution using hyper-laplacian priors. In *NIPS*, 2009.

[10] A. Levin, R. Fergus, R. Fergus, F. Durand, and W. T. Freeman. Image and depth from a conventional camera with a coded aperture. In *SIGGRAPH*, 2007.

[11] A. Levin, P. Sand, T. S. Cho, F. Durand, and W. T. Freeman. Motion-invariant photography. In *SIGGRAPH*, 2008.

[12] S. McCloskey. Velocity-dependent shutter sequences for motion deblurring. In *European Conf. on Computer Vision*, 2010.

[13] S. McCloskey, W. Au, and J. Jelinek. Iris capture from moving subjects using a fluttering shutter. In *Proc. of the Fourth IEEE Conference on Biometrics: Theory, Applications, and Systems*, 2010.

[14] S. McCloskey, Y. Ding, and J. Yu. Design and estimation of coded exposure point spread functions. *IEEE Trans. Pattern*

Analysis and Machine Intelligence, 34(10):2071–2077, Oct. 2012.

[15] S. McCloskey, K. Muldoon, and S. Venkatesha. Motion invariance and custom blur from lens motion. In *Int'l Conf. on Computational Photography*, 2011.

[16] H. Nagahara, S. Kuthirummal, C. Zhou, and S. Nayar. Flexible Depth of Field Photography. In *European Conf. on Computer Vision*, 2008.

[17] R. Raskar, A. Agrawal, and J. Tumblin. Coded exposure photography: motion deblurring using fluttered shutter. *ACM Trans. on Graphics*, 25(3):795–804, 2006.

[18] T. Sonoda, H. Nagahara, and R. Taniguchi. Motion-invariant coding using a programmable aperture camera. In *Asian Conf. on Computer Vision*, 11 2012.

[19] A. Veeraraghavan, R. Raskar, A. Agrawal, A. Mohan, and J. Tumblin. Dappled photography: Mask enhanced cameras for heterodyned light fields and coded aperture refocusing. In *SIGGRAPH*, 2007.

[20] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce. Non-uniform deblurring for shaken images. *Int'l Journal of Computer Vision*, 98(2):168–186, 2012.

[21] W. Xu and S. McCloskey. 2d barcode localization and motion deblurring using a flutter shutter camera. In *Workshop on Applications of Computer Vision*, 2011.

[22] C. Zhou and S. K. Nayar. Computational cameras: Convergence of optics and processing. *IEEE Transactions on Image Processing*, 20(12):3322–3340, 2011.