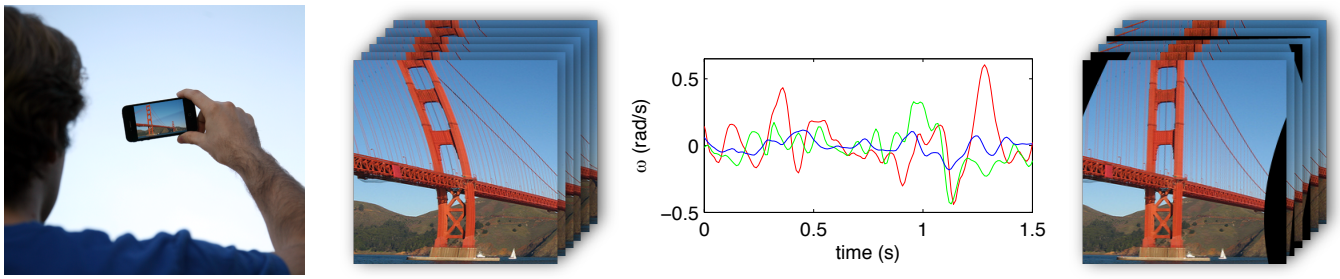
Digital Video Stabilization and Rolling Shutter Correction using Gyroscopes

Alexandre Karpenko
Stanford University

David Jacobs
Stanford University

Jongmin Baek
Stanford University

Marc Levoy
Stanford University



(a) **(b)** **(c)** **(d)**

Figure 1: *(a) Videos captured with a cell-phone camera tend to be shaky due to the device's size and weight. (b) The rolling shutter used by sensors in these cameras also produces warping in the output frames (we have exagerrated the effect for illustrative purposes). (c) We use gyroscopes to measure the camera's rotations during video capture. (d) We use the measured camera motion to stabilize the video and to rectify the rolling shutter. (Golden Gate photo courtesy of Salim Virji.)*

Abstract

In this paper we present a robust, real-time video stabilization and rolling shutter correction technique based on commodity gyroscopes. First, we develop a unified algorithm for modeling camera motion and rolling shutter warping. We then present a novel framework for automatically calibrating the gyroscope and camera outputs from a single video capture. This calibration allows us to use only gyroscope data to effectively correct rolling shutter warping and to stabilize the video. Using our algorithm, we show results for videos featuring large moving foreground objects, parallax, and low-illumination. We also compare our method with commercial image-based stabilization algorithms. We find that our solution is more robust and computationally inexpensive. Finally, we implement our algorithm directly on a mobile phone. We demonstrate that by using the phone's inbuilt gyroscope and GPU, we can remove camera shake and rolling shutter artifacts in real-time.

CR Categories: I.4.3 [Computing Methodologies]: Image Processing and Computer Vision—Enhancement; I.4.1 [Computing Methodologies]: Image Processing and Computer Vision—Digitization and Image Capture

Keywords: video stabilization, rolling shutter correction, gyroscopes, mobile devices

1 Introduction

Digital still cameras capable of capturing video have become widespread in recent years. While the resolution and image quality of these consumer devices has improved up to the point where they rival DSLRs in some settings, their video quality is still significantly worse than that of film cameras. The reason for this gap in quality is twofold. First, compared to film cameras, cell phones are significantly lighter. As a result, hand-held video capture on such devices exhibits a greater amount of camera shake. Second, most cell-phone cameras have sensors that make use of a *rolling shutter* (RS). In an RS camera, each image row is exposed at a slightly different time; which, combined with undampened camera motion, results in a nauseating "wobble" in the output video.

In the following sections we present our technique for improving the video quality of RS cameras. Specifically, we employ inexpensive microelectromechanical (MEMS) gyroscopes to measure camera rotations. We use these measurements to perform video stabilization (inter-frame motion compensation) and rolling shutter correction (intra-frame motion compensation). To our knowledge, we are the first to present a gyroscope-based solution for digital video stabilization and rolling shutter correction. Our approach is both computationally inexpensive and robust. This makes it particularly suitable for real-time implementations on mobile platforms.

Our technique is based on a unified model of a rotating camera and a rolling shutter. We show how this model can be used to compute a warp that simultaneously performs rolling shutter correction and video stabilization. We also develop an optimization framework that automatically calibrates the gyroscope and camera. This allows us to recover unknown parameters such as gyroscope drift and delay, as well as the camera's focal length and rolling shutter speed from a single video and gyro capture. As a result any combination of gyroscope and camera hardware can be calibrated without the need for a specialized laboratory setup.

Finally, we demonstrate the practicality of our approach by implementing real-time video stabilization and rolling shutter correction on Apple's iPhone 4.

1.1 Related Work

Video stabilization is a family of techniques used to reduce high-frequency frame-to-frame jitter produced by video camera shake. In professional cameras, mechanical image stabilization (MIS) sys-

tems are commonly used. For example, in the SteadiCam system the operator wears a harness that separates the camera's motion from the operator's body motion. Other MIS systems stabilize the optics of the camera rather than the camera body itself. These systems work by moving the lens or sensor to compensate for small pitch and yaw motions. These techniques work in real time and do not require computation on the camera. However, they are not suitable for mobile devices and inexpensive cameras, because of their price and size.

As a result, a number of digital stabilization systems have been developed that stabilize videos post-capture. Digital video stabilization typically employs feature trackers to recover image-plane (2D) motion [Matsushita et al. 2006; Battiato et al. 2007] or to extract the underlying (3D) camera motion [Buehler et al. 2001; Bhat et al. 2007; Liu et al. 2009]. A low-pass filter is applied to the recovered motion, and a new video is generated by synthesizing frames along this smoothed path. However, feature trackers are sensitive to noise (such as fast moving foreground objects) and require distinctive features for tracking. As a result, digital stabilization based on feature tracking often breaks down—especially in adverse lighting conditions and excessive foreground motion. In addition, extracting and matching visual cues across frames is computationally expensive, and that expense grows with the resolution of the video. This becomes prohibitively costly for some algorithms if the goal is to perform video stabilization in real time. Consequently, such approaches are rarely employed in current digital cameras. Instead, manufacturers opt for more robust (and expensive) mechanical stabilization solutions for high-end DSLRs.

Rolling shutter correction is a related family of techniques for removing image warping produced by intra-frame camera motion. High-end cameras use CCD sensors, which have a *global shutter* (GS). In a GS camera (including many DSLRs) all pixels on the CCD sensor are read out and reset simultaneously. Therefore all pixels collect light during the same time interval. Consequently, camera motion during the exposure results in some amount of image blur on these devices. In contrast, low-end cameras typically make use of CMOS sensors. In particular, these sensors employ a rolling shutter, where image rows are read out and reset sequentially. The advantage of this approach is that it requires less circuitry compared to CCD sensors. This makes CMOS sensors cheaper to manufacture [El Gamal and Eltoukhy 2005]. For that reason, CMOS sensors are frequently used in cell phones, music players, and some low-end camcorders [Forssén and Ringaby 2010]. The sequential readout, however, means that each row is exposed during a slightly different time window. As a result, camera motion during row readout will produce a warped image. Fast moving objects will also appear distorted.

Image readout in an RS camera is typically in the millisecond range. Therefore, RS distortions are primarily caused by high-frequency camera motions. MIS systems could, therefore, be used to stabilize the camera. While this approach removes rolling shutter warping, in practice the price range and size of MIS systems make it not suitable for RS cameras. For that reason, a number of digital rolling shutter rectification techniques have been developed. Ait-Aider et al. [2007] develop a technique for correcting RS artifacts in a single image. Our approach also works for single images, but unlike Ait-Aider et al.'s method, it does not require user input. However, in this paper we restrict our analysis to videos. A number of techniques have been proposed for rectifying RS in a sequence of frames [Cho and Hong 2007; Liang et al. 2008; Forssén and Ringaby 2010]. Forssén and Ringaby [2010] use feature tracking to estimate the camera motion from the video. Once the camera motion is known during an RS exposure, it can be used to rectify the frame. Since this approach relies on feature trackers, it has the same disadvantages previously discussed in the case of video stabilization.

Our approach foregoes the use of feature trackers or MIS systems. Instead, we employ inexpensive MEMS gyroscopes to measure camera motion directly. Inertial measurement units (IMUs) have been successfully used for image de-blurring [Joshi et al. 2010] and for aiding a KLT feature tracker [Hwangbo et al. 2009]. They are also frequently used for localization and mechanical stabilization in robotics [Kurazume and Hirose 2000].

Measuring camera motion using gyroscopes allows us to perform digital video stabilization and RS rectification with high computational efficiency. This approach is robust even under poor lighting or substantial foreground motion, because we do not use the video's content for motion estimation. While our method requires an additional hardware component, many current camera-enabled mobile phones—such as the iPhone 4—are already equipped with such a device. Furthermore, compared to MIS systems, MEMS gyroscopes are inexpensive, versatile and less bulky (see fig. 8). We believe that our approach strikes a good balance between computational efficiency, robustness, size and price range for the large market of compact consumer cameras and cell phone cameras.

2 Video Stabilization and Rolling Shutter Correction

Video stabilization typically proceeds in three stages: camera motion estimation, motion smoothing, and image warping. Rolling shutter rectification proceeds in the same way; except the actual camera motion is used for the warping computation rather than the smoothed motion. As we will later show, both video stabilization and rolling shutter correction can be performed in one warping computation under a unified framework. We develop this framework in the following subsections.

We begin by introducing a model for an RS camera and its motion. This model is based on the work presented by Forssén and Ringaby [2010]. Forssén and Ringaby use this RS camera model in conjunction with a feature tracker to rectify rolling shutter in videos. The reliance on feature trackers, however, makes their system susceptible to the same issues as tracker-based video stabilization algorithms. We extend their model to a unified framework that can perform both rolling shutter correction and video stabilization in one step. We also develop an optimization procedure that allows us to automatically recover all the unknowns in our model from a single input video and gyroscope recording.

Camera motion in our system is modeled in terms of rotations only. We ignore translations because they are difficult to measure accurately using IMUs. Also, accelerometer data must be integrated twice to obtain translations. In contrast, gyroscopes measure the rate of rotation. Therefore, gyro data needs to be integrated only once to obtain the camera's orientation. As a result, translation measurements are significantly less accurate than orientation measurements [Joshi et al. 2010]. Even if we could measure translations accurately, this is not sufficient since objects at different depths move by different amounts. Therefore, we would have to rely on stereo or feature-based structure from motion (SfM) algorithms to obtain depth information. Warping frames in order to remove translations is non-trivial due to parallax and occlusions. These approaches are not robust and are currently too computationally expensive to run in real time on a mobile platform.

Forssén and Ringaby [2010] have attempted to model camera translations in their system; but found the results to perform worse than a model that takes only rotations into account. They hypothesize that their optimizer falls into a local minimum while attempting to reconstruct translations from the feature tracker. Their algorithm also assumes that the camera is imaging a purely planar scene (i.e.,

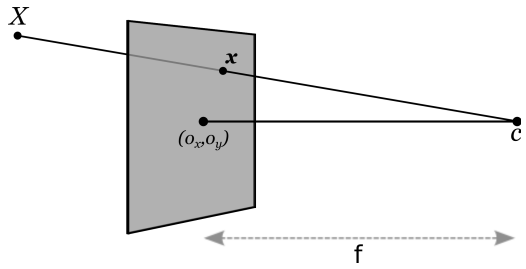


Figure 2: *Pinhole camera model. A ray from the camera center* $\mathbf{c}$ *to a point in the scene* $\mathbf{X}$ *will intersect the image plane at* $\mathbf{x}$*. Therefore the projection of the world onto the image plane depends on the camera's center* $\mathbf{c}$*, the focal length* f*, and the location of the camera's axis* (o_x, o_y) *in the image plane.*

constant depth). Therefore, translation reconstruction sometimes fails due to unmodeled parallax in the video.

To avoid these problems we do not incorporate translations into our model. Fortunately, camera shake and rolling shutter warping occur primarily from rotations. This is the case because translations attenuate quickly with increasing depth, and objects are typically sufficiently far away from the lens that translational camera jitter does not produce noticeable motion in the image. This conclusion is supported by our stabilization results.

2.1 Camera Model

Our rotational rolling shutter camera model is based on the pinhole camera model. In a pinhole camera the relationship between image point $\mathbf{x}$ in homogeneous coordinates and the corresponding point $\mathbf{X}$ in 3D world coordinates (fig. 2) may be specified by:

$$\mathbf{x} = \mathbf{K}\mathbf{X}\,, \text{ and } \quad \mathbf{X} = \lambda \mathbf{K}^{-1}\mathbf{x}. \tag{1}$$

Here, λ is an unknown scaling factor and $\mathbf{K}$ is the intrinsic camera matrix, which we assume has an inverse of the following form:

$$\mathbf{K}^{-1} = \begin{pmatrix} 1 & 0 & -o_x \\ 0 & 1 & -o_y \\ 0 & 0 & f \end{pmatrix}, \tag{2}$$

where, (o_x, o_y) is the origin of the camera axis in the image plane and f is the focal length. The camera's focal length is an unknown that we need to recover. We assume that the camera has square pixels by setting the upper diagonal entries to 1. However, it is straightforward to extend this model to take into account non-square pixels or other optical distortions.

2.2 Camera Motion

We set the world origin to be the camera origin. The camera motion can then be described in terms of its orientation $\mathbf{R}(t)$ at time t. Thus, for any scene point $\mathbf{X}$, the corresponding image point $\mathbf{x}$ at time t is given by:

$$\mathbf{x} = \mathbf{K}\mathbf{R}(t)\mathbf{X}. \tag{3}$$

The rotation matrices $\mathbf{R}(t) \in \mathrm{SO}(3)$ are computed by compounding the changes in camera angle $\Delta\theta(t)$. We use SLERP (Spherical Linear intERPolation) of quaternions [Shoemake 1985] in order to interpolate the camera orientation smoothly and to avoid gimbal lock.[1] $\Delta\theta(t)$ is obtained directly from gyroscope measured rates of rotation $\omega(t)$:

$$\Delta\theta(t) = (\omega(t + t_d) + \omega_d) * \Delta t. \tag{4}$$

Here ω_d is the gyroscope drift and t_d is the delay between the gryoscope and frame sample timestamps. These parameters are additional unknowns in our model that we also need to recover.

[1] In practice, the change in angle between gyroscope samples is sufficiently small that Euler angles work as well as rotation quaternions.

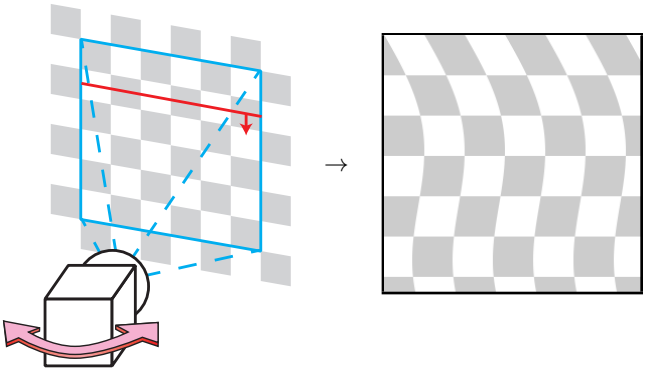


Figure 3: *High-frequency camera rotations while the shutter is rolling from top to bottom cause the output image to appear warped.*

2.3 Rolling Shutter Compensation

We now introduce the notion of a rolling shutter into our camera model. Recall that in an RS camera each image row is exposed at a slightly different time. Camera rotations during this exposure will, therefore, determine the warping of the image.[2] For example, if the camera sways from side to side while the shutter is rolling, then the output image will be warped as shown in fig. 3. The time at which point $\mathbf{x}$ was imaged in frame i depends on how far down the frame it is. More formally, we can say that $\mathbf{x}$ was imaged at time $t(i, y)$:

$$t(i, y) = t_i + t_s * y/h\,, \quad \text{where } \mathbf{x} = (x, y, 1)^T, \tag{5}$$

where y is the image row corresponding to point $\mathbf{x}$, h is the total number of rows in the frame, and t_i is the timestamp of the i-th frame. The t_s term states that the farther down we are in a frame, the longer it took for the rolling shutter to get to that row. Hence, t_s is the time it takes to read out a full frame going row by row from top to bottom. Note that a negative t_s value would indicate a rolling shutter that goes from bottom to top. We will show how to automatically recover the sign and value of t_s in section 3.

2.4 Image Warping

We now derive the relationship between image points in a pair of frames for two different camera orientations (see fig. 4). For a scene point $\mathbf{X}$ the projected points $\mathbf{x}_i$ and $\mathbf{x}_j$ in the image plane of two frames i and j are given by:

$$\mathbf{x}_i = \mathbf{K}\mathbf{R}(t(i, y_i))\mathbf{X}\,, \text{ and } \quad \mathbf{x}_j = \mathbf{K}\mathbf{R}(t(j, y_j))\mathbf{X}. \tag{6}$$

If we rearrange these equations and substitute for $\mathbf{X}$, we get a mapping of all points in frame i to all points in frame j:

$$\mathbf{x}_j = \mathbf{K}\mathbf{R}(t(j, y_j))\mathbf{R}^{\mathbf{T}}(t(i, y_i))\mathbf{K}^{-1}\mathbf{x}_i\,. \tag{7}$$

[2] Translational camera jitter during rolling shutter exposure does not significantly impact image warping, because objects are typically far away from the lens.

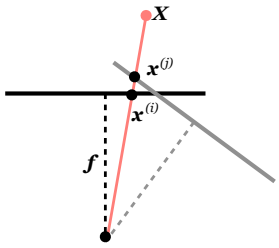


Figure 4: *Top view of two camera orientations and their corresponding image planes i and j. An image of scene point $\mathbf{X}$ appears in the two frames where the ray (red) intersects their camera plane.*

So far we have considered the relationship between two frames of the same video. We can relax this restriction by mapping frames from one camera that rotates according to $\mathbf{R}(t)$ to another camera that rotates according to $\mathbf{R}'(t)$. Note that we assume both camera centers are at the origin. We can now define the warping matrix $\mathbf{W}$ that maps points from one camera to the other:

$$\mathbf{W}(t_1, t_2) = \mathbf{K}\mathbf{R}'(t_1)\mathbf{R}^{\mathbf{T}}(t_2)\mathbf{K}^{-\mathbf{1}}. \quad (8)$$

Notice that eq. 7 can now be expressed more compactly as:

$$\mathbf{x}_j = \mathbf{W}(t(j, y_j), t(i, y_i))\mathbf{x}_i \ , \ \text{where} \ \ \mathbf{R}' = \mathbf{R}. \quad (9)$$

Also note that $\mathbf{W}$ depends on both image rows y_i and y_j of image points $\mathbf{x}_i$ and $\mathbf{x}_j$ respectively. This warping matrix can be used match points in frame i to corresponding points in frame j, while taking the effects of the rolling shutter into account in both frames.

Given this formulation of a warping matrix, the algorithm for rolling shutter correction and video stabilization becomes simple. We create a synthetic camera that has a smooth motion and a global shutter. This camera's motion is computed by applying a Gaussian low-pass filter to the input camera's motion, which results in a new set of rotations $\mathbf{R}'$. We set the rolling shutter duration t_s for the synthetic camera to 0 (i.e., a global shutter). We then compute $\mathbf{W}(t_i, t(i, y_i))$ at each image row y_i of the current frame i, and apply the warp to that row. Notice that the first term of $\mathbf{W}$ now only depends on the frame time t_i. This operation maps all input frames onto our synthetic camera; and as a result, simultaneously removes rolling shutter warping and video shake.

In practice, we do not compute $\mathbf{W}(t_i, t(i, y_i))$ for each image row y_i. Instead, we subdivide the input image (fig. 5a) and compute the warp at each vertical subdivision (fig. 5c and 5d). In essence, we create a warped mesh from the input image that is a piecewise linear approximation of the non-linear warp. We find that ten subdivisions are typically sufficient to remove any visible RS artifacts. Forssén and Ringaby [2010] refer to this sampling approach as *inverse interpolation*. They also propose two additional interpolation techniques, which they show empirically to perform better on a synthetic video dataset. However, we use inverse interpolation because it is easy to implement an efficient version on the GPU using vertex shaders. The GPU's fragment shader takes care of resampling the mesh-warped image using bilinear interpolation. We find that RS warping in actual videos is typically not strong enough to produce aliasing artifacts due to bilinear inverse interpolation. As a result, inverse interpolation works well in practice.

Some prior work in rolling shutter correction makes use of global image warps—such as the global affine model [Liang et al. 2008] and the global shift model [Chun et al. 2008]. These models assume that camera rotation is more or less constant during rolling shutter exposure. If this is not the case, then a linear approximation will fail to rectify the rolling shutter (fig. 5b). We evaluate the performance of a linear approximation on actual video footage in section 4.

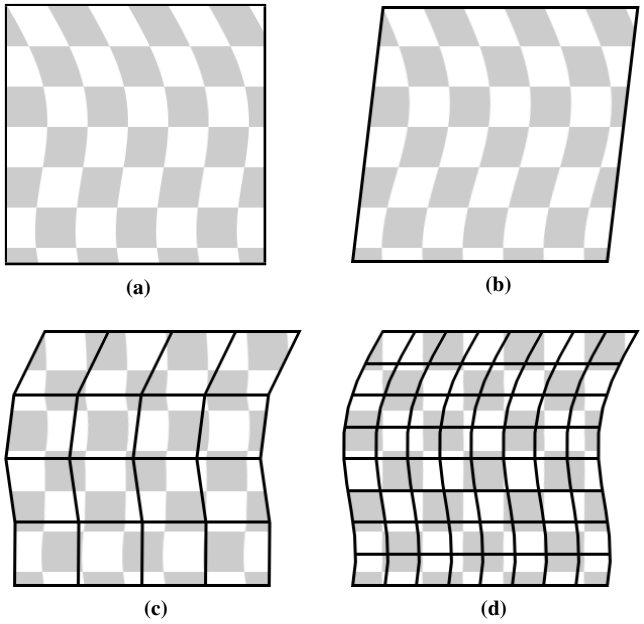


Figure 5: *(a) Warped image captured by an RS camera. (b) A global linear transformation of the image, such as the shear shown here, cannot fully rectify the warp. (c) We use a piecewise linear approximation of non-linear warping. (d) We find that 10 subdivisions are sufficient to eliminate visual artifacts.*

3 Camera and Gyroscope Calibration

We now present our framework for recovering the unknown camera and gyroscope parameters. This calibration step is necessary to enable us to compute $\mathbf{W}$ directly from the gyroscope data. The unknown parameters in our model are: the focal length of the camera f, the duration of the rolling shutter t_s, the delay between the gyroscope and frame sample timestamps t_d, and the gyroscope drift w_d.

Note that some of these parameters, such as the camera's focal length, might be specified by the manufacturer. It is alternatively possible to measure these parameters experimentally. For example, Forssén and Ringaby [2010] use a quickly flashing display to measure the rolling shutter duration t_s. However, these techniques tend to be imprecise and error prone; and they are also too tedious to be carried out by regular users. The duration of the rolling shutter is typically in the millisecond range. As a result, a small misalignment in t_d or t_s would cause rolling shutter rectification to fail.

Our approach is to estimate these parameters from a single video and gyroscope capture. The user is asked to record a video and gyroscope trace where they stand still and shake the camera while pointing at a building. A short clip of about ten seconds in duration is generally sufficient to estimate all the unknowns. Note that this only needs to be done once for each camera and gyroscope arrangement.

In our approach, we find matching points in consecutive video

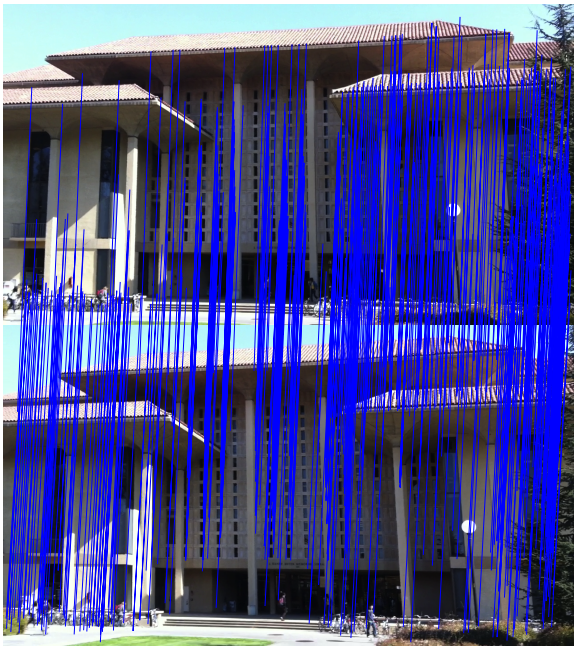


Figure 6: *Point correspondences in consecutive frames. We use SIFT to find potential matches. We then apply RANSAC to discard outliers that do not match the estimated homography.*

frames using SIFT [Lowe 2004], and we use RANSAC [Fischler and Bolles 1981] to discard outliers. The result is a set of point correspondences $\mathbf{x}_i$ and $\mathbf{x}_j$ for all neighboring frames in the captured video (fig. 6). Given this ground truth, one can formulate calibration as an optimization problem, where we want to minimize the mean-squared re-projection error of all point correspondences:

$$J = \sum_{(i,j)} ||\mathbf{x}_j - \mathbf{W}(t(j, y_j), t(i, y_i))\mathbf{x}_i||^2. \qquad (10)$$

Note that this is a non-linear optimization problem. A number of non-linear optimizers could be used to minimize our objective function. However, we have found coordinate descent by direct objective function evaluation to converge quickly. Each time we take a step where the objective function J does not decrease, we reverse the step direction and decrease the step size of the corresponding parameter. The algorithm terminates as soon as the step size for all parameters drops below a desired threshold (i.e., when we have achieved a target precision). Our Matlab/C++ implementation typically converges in under 2 seconds for a calibration video of about 10 seconds in duration.

We initialize our optimization algorithm by setting the focal length to be such that the camera has a field of view of 45°. We set all other parameters to 0. We find that with these initial conditions, the optimizer converges to the correct solution for our dataset. More generally, we can avoid falling into a local minimum (e.g., when the delay between the gyro and frame timestamps is large) by restarting our coordinate descent algorithm for a range of plausible parameters, and selecting the best solution. The average re-projection error for correctly recovered parameters is typically around 1 pixel.

An additional unknown in our model is the relative orientation of the gyroscope to the camera. For example, rotations about the gyro's y-axis could correspond to rotations about the camera's x-axis. To discover the gyroscope orientation we permute its 3 rotation axes and run our optimizer for each permutation. The permutation that minimizes the objective best corresponds to the camera's axis ordering. We found re-projection error to be significantly larger for incorrect permutations. Therefore, this approach works well in practice.

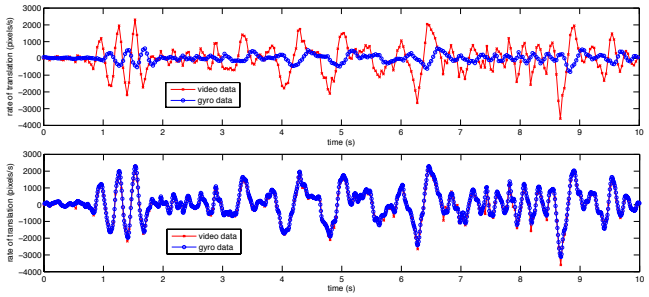


Figure 7: *Signals* $\mathbf{x}$ *(red) and* $f * \omega_y(t + t_d)$ *(blue). Top: Before calibration the amplitude of the signals does not match, because our initial guess for* f *is too low. In addition, the signals are shifted since we initialize* t_d *to 0. Bottom: After calibration the signals are well aligned because we have recovered accurate focal length and gyroscope delay.*

In our discussion we have assumed that the camera has a vertical rolling shutter. The RS model could be easily modified to work for image columns instead of rows. Finding the minimum re-projection error for both cases would tell us whether the camera has a horizontal or vertical rolling shutter.

Finally, in order to provide a better sense of the results achieved by calibration, we present a visualization of video and gyroscope signals before and after calibration. If we assume that rotations between consecutive frames are small, then translations in the image can be approximately computed from rotations as follows:

$$\dot{\mathbf{x}}(t) \approx f * \hat{\omega}(t + t_d), \text{ where } \begin{cases} \dot{\mathbf{x}} = (\dot{x}, \dot{y})^T \\ \hat{\omega} = (\omega_y, \omega_x)^T \end{cases} \qquad (11)$$

Here, we have also assumed no effects due to rolling shutter (i.e., $t_s = 0$), and we ignore rotations about the z-axis (i.e., ω_z). We let $\dot{\mathbf{x}}$ be the average rate of translation along x and y for all point correspondences in consecutive frames. If our optimizer converged to the correct focal length f and gyro delay t_d, then the two signals should align. Fig. 7 plots the first dimension of signals $\dot{\mathbf{x}}$ and $f * \hat{\omega}(t + t_d)$ before and after alignment. Note how accurately the gyroscope data matches the image motions. This surprising precision of MEMS gyroscopes is what enables our method to perform well on the video stabilization and rolling shutter correction tasks.

4 Results

In this section we present dataset and results for video stabilization and rolling shutter correction. We also compare our approach with a number of feature tracker based algorithms.

4.1 Video and Gyroscope Dataset

We use an iPhone 4 to capture video and gyroscope data. The platform has a MEMS gyroscope (see fig. 8), which we run at a (maximum) frequency of 100Hz. Furthermore, the phone has an RS camera capable of capturing 720p video at 30 frames per second (fps). The frame-rate is variable; and typically adjusts in low-illumination settings to 24fps. We record the frame timestamps as well as the

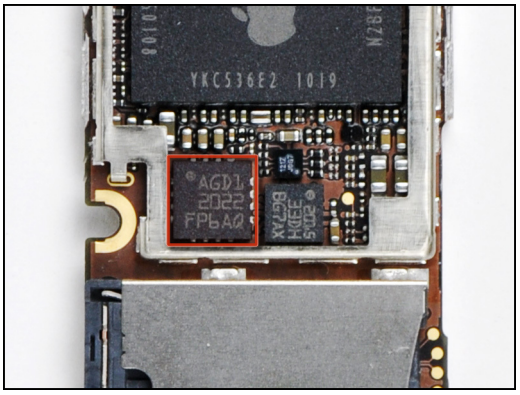


Figure 8: *The iPhone 4's MEMS gyroscope (outlined in red). (Photo courtesy of iFixit.com.)*

timestamped gyroscope data, which are saved along with the captured video.

Our aim was to obtain a wide range of typical videos captured by non-professionals. We have recorded videos where the camera is moving and videos where the camera is mostly stationary. Videos in our dataset also contain varying amounts of moving foreground objects and illumination conditions. We also record a calibration video, which we use to recover the iPhone's camera and gyroscope parameters. Except for the calibration video, the video shake in our videos was never deliberate—it is simply a consequence of the device being very light.

4.2 Evaluation

We ask the reader to refer to the accompanying videos in order to obtain a better sense of our results. We have included four handheld video sequences from our test dataset: the first sequence contains a walking motion, the second features a strong lens flare, the third contains cars moving in the foreground and the fourth sequence was captured at night. In addition we provide the calibration video used to recover camera and gyroscope parameters. For the walking sequence we also include two additional videos. The first shows the wobble that occurs when RS compensation is turned off. The second shows RS correction results, in which warping is approximated with a global homography.

[Note to reviewers: the videos included in our submission are a subset of our test dataset, which will be made available online to accompany the published paper. It was impossible to provide a link to the full dataset without compromising anonymity.]

We compare our stabilization and RS correction results with image-based video stabilization solutions. We use iMovie'11 and Deshaker to stabilize our videos. Both applications offer rolling shutter correction. We find that iMovie'11 and Deshaker produce subpar results for most videos in our dataset. Frame popping and jitter from failed RS compensation can be seen in each of the four videos. In contrast, our method performs well regardless of video content.

Although our model does not compensate for translations, high-frequency translational jitter is not visible in our output videos. This supports our original conclusion that camera shake and rolling shutter warping occurs primarily due to rotations. A low-frequency translational up and down motion can be seen in the stabilized walking sequence that corresponds to the steps taken by the camera's user.

One of our experimental results is the observation that if one accurately stabilizes a video but does not correct for rolling shutter, and the original video contains high-frequency camera rotations, then the stabilized video will look poor. In effect, correcting accurately for one artifact makes the remaining artifact more evident and egregious. To support this observation, our dataset includes an example where we have disabled rolling shutter rectification. We also find that a linear approximation for RS warping is not sufficient to completely remove RS artifacts in more pronounced cases (e.g., at each step in the walking motion). We have included a video where rolling shutter warping is rectified with a global homography. In this video, artifacts due to warping non-linearities are still clearly visible. As a result, our algorithm performs better than linear RS approximations such as Liang et al. [2008] and Chun et al. [2008].

Apart from scenes where feature tracking fails, 2D stabilization algorithms also conflate translations that occur due to parallax with translations that occur due to the camera's rotation. This degrades the accuracy of the recovered 2D camera motion in the presence of parallax. As a result, frame popping and jitter can be seen in many videos produced by iMovie and Deshaker. In addition, high-frequency camera motions are difficult to reconstruct in the presence of noise. Therefore, rolling shutter correction is a difficult task for feature-based algorithms. Our approach, on the other hand, is effective at correcting RS artifacts because gyroscopes measure the camera's rotation with high frequency and high accuracy.

Deshaker and iMovie are 2D stabilization solutions that reconstruct 2D motion in the image plane. Our method is also a 2D stabilization algorithm, because we do not measure the camera's translation. In contrast, 3D stabilization algorithms recover the camera's full 3D motion. However, they rely on structure from motion (SfM) techniques that are currently more brittle than 2D tracking. For example, Liu et al. [2009] use Voodoo to reconstruct 3D camera motion and a feature point cloud. They use this reconstruction to perform 3D video stabilization using content preserving warps. However, we find that Voodoo fails to correctly recover the 3D structure and camera motion in many of the videos in our dataset (e.g., the video captured at night).

We have found motion blur in low-illumination videos (e.g., the night sequence) to significantly degrade the quality of our stabilization results. While our algorithm performs better than feature-based stabilization on the night sequence, motion blur from the original shaky camera video is clearly visible in the stabilized output. However, removing this artifact is out of the scope of this paper.

Finally, our method can be easily used for RS correction in single high-resolution photographs since our algorithm already works for individual video frames. Ait-Aider et al. [2007] looked at rectifying RS post-capture in single images. However, unlike their approach we do not require any user input. We leave a more detailed analysis of this application for future work.

4.3 Realtime Implementation

To demonstrate the low computational expense of our approach, we have implemented our method to run in real time on the iPhone 4. Using our algorithm and the built-in gyroscope, we are able to display a stabilized and rolling shutter corrected viewfinder directly on the iPhone's screen. Our implementation runs at 30 fps (i.e., the camera's maximum frame rate).

We receive frames from the camera and copy them to the GPU, where we perform the warping computation using vertex shaders and a subdivided textured mesh (as described in section 2.4). Moving frames to the GPU is the bottleneck in this approach; however, we found this to be substantially faster than performing warping computations on the CPU, even though the latter avoids extra frame

copies.

In order to prevent a large delay in the viewfinder, we use a truncated causal low-pass filter for computing smooth output rotations. Compared to the Gaussian filter used in the previous sections, this causal filter attenuates camera shake but does not completely eliminate it. However, RS correction is unaffected by this filter change, because it is computed from the unsmoothed rotations during the frame's exposure period.

For video recording, frames can be held back for a longer period of time before they need to be passed off to the video encoder. As a result, a better low-pass filter can be used than in the case of a viewfinder, which must display imagery with low latency. We leave the implementation of such a recording pipeline for future work.

5 Conclusion

In this paper, we have presented an algorithm that employs gyroscopes for digital video stabilization and rolling shutter correction. We have developed an optimization framework that can calibrate the camera and gyroscope data from a single input video. In addition, we have demonstrated that MEMS gyroscopes are sufficiently accurate to successfully stabilize video and to correct for rolling shutter warping. We have compared our approach to video stabilization based on feature tracking. We have found that our approach is more efficient and more robust in a diverse set of videos.

The main limitation of our method is that it is restricted to rotations only. While this makes our approach robust and computationally efficient, 3D video stabilization can produce better results when a specific camera translation is desired. For example, Forssén and Ringaby's [2010] present a 3D video stabilization algorithm that can synthesize a dolly shot (i.e., camera motion along a straight line) from hand-held video. Future work could investigate combining IMUs and feature trackers in order to improve the accuracy and robustness of the reconstructed camera motion.

Another limitation of frame warping is that it produces areas for which there is no image data. We crop video frames in order to hide these empty areas. This operation reduces the field of view of the camera and also discards video data around frame boundaries. Future work could investigate using inpainting algorithms [Matsushita et al. 2006] to perform full-frame stabilization.

Lastly, we do not currently remove motion blur. This degrades the quality of stabilized low-illumination videos in our dataset. Joshi et al. [2010] have presented an effective IMU aided image deblurring algorithm. Their approach fits in well with our method since both algorithms rely on gyroscopes. Alternatively, future work could explore the use of alternating consecutive frame exposures for inverting motion blur in videos [Agrawal et al. 2009].

References

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

AIT-AIDER, O., BARTOLI, A., AND ANDREFF, N. 2007. Kinematics from lines in a single rolling shutter image. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, 1–6.

BATTIATO, S., GALLO, G., PUGLISI, G., AND SCELLATO, S. 2007. SIFT features tracking for video stabilization. *Image Analysis and Processing, International Conference on 0*, 825–830.

BHAT, P., ZITNICK, C. L., SNAVELY, N., AGARWALA, A., AGRAWALA, M., CURLESS, B., COHEN, M., AND KANG, S. B. 2007. Using photographs to enhance videos of a static scene. In *Rendering Techniques 2007 (Proceedings Eurographics Symposium on Rendering)*, J. Kautz and S. Pattanaik, Eds., Eurographics, 327–338.

BUEHLER, C., BOSSE, M., AND MCMILLAN, L. 2001. Non-metric image-based rendering for video stabilization. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on 2*, 609.

CHO, W., AND HONG, K.-S. 2007. Affine motion based CMOS distortion analysis and cmos digital image stabilization. *Consumer Electronics, IEEE Transactions on 53*, 3, 833 –841.

CHUN, J.-B., JUNG, H., AND KYUNG, C.-M. 2008. Suppressing rolling-shutter distortion of cmos image sensors by motion vector detection. *Consumer Electronics, IEEE Transactions on 54*, 4, 1479 –1487.

EL GAMAL, A., AND ELTOUKHY, H. 2005. Cmos image sensors. *Circuits and Devices Magazine, IEEE 21*, 3, 6 – 20.

FISCHLER, M. A., AND BOLLES, R. C. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24* (June), 381–395.

FORSSÉN, P.-E., AND RINGABY, E. 2010. Rectifying rolling shutter video from hand-held devices. In *CVPR*, 507–514.

HWANGBO, M., KIM, J.-S., AND KANADE, T. 2009. Inertial-aided klt feature tracking for a moving camera. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, 1909 –1916.

JOSHI, N., KANG, S. B., ZITNICK, C. L., AND SZELISKI, R. 2010. Image deblurring using inertial measurement sensors. *ACM Trans. Graph. 29* (July), 30:1–30:9.

KURAZUME, R., AND HIROSE, S. 2000. Development of image stabilization system for remote operation of walking robots. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*.

LIANG, C.-K., CHANG, L.-W., AND CHEN, H. 2008. Analysis and compensation of rolling shutter effect. *Image Processing, IEEE Transactions on 17*, 8, 1323 –1330.

LIU, F., GLEICHER, M., JIN, H., AND AGARWALA, A. 2009. Content-preserving warps for 3d video stabilization. *ACM Trans. Graph. 28* (July), 44:1–44:9.

LOWE, D. G. 2004. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision 60* (November), 91–110.

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

SHOEMAKE, K. 1985. Animating rotation with quaternion curves. *SIGGRAPH Comput. Graph. 19* (July), 245–254.