Removing Rolling Shutter Wobble

Simon Baker, Eric Bennett, Sing Bing Kang, and Richard Szeliski

Microsoft Corporation
One Microsoft Way
Redmond WA 98052

Abstract

We present an algorithm to remove wobble artifacts from a video captured with a rolling shutter camera undergoing large accelerations or jitter. We show how estimating the rapid motion of the camera can be posed as a temporal super-resolution problem. The low-frequency measurements are the motions of pixels from one frame to the next. These measurements are modeled as temporal integrals of the underlying high-frequency jitter of the camera. The estimated high-frequency motion of the camera is then used to re-render the sequence as though all the pixels in each frame were imaged at the same time. We also present an auto-calibration algorithm that can estimate the time between the capture of subsequent rows in the camera.

1. Introduction

Most digital still cameras, cellphone cameras, and webcams use CMOS sensors. CMOS video cameras are also increasingly becoming popular, from the low-end Flip camera [12] to the high-end Red camera [13]. To maximize the fill factor, CMOS sensors are commonly read out line-by-line and use a "rolling shutter"; ie. the capture time of each row is slightly after the capture time of the previous row.

Rolling shutter cameras suffer from three main artifacts: (1) shear, (2) partial exposure, and (3) wobble. Shearing occurs when the camera undergoes a constant (or smoothly varying) motion. Shearing can be corrected by computing the global motion and then warping the frames appropriately [11, 7]. In the presence of independently moving (but slowly accelerating) objects, a full optical flow field can be used to perform the correction [5]. Partial exposure occurs when a rolling shutter is used to image fast changing illumination such as a flash, a strobe light, or lightning [5].

Wobble occurs when there are large accelerations or the motion is at a higher frequency than the frame rate of the camera. Wobble is particularly pronounced for cameras mounted on helicopters, cars, and motorbikes. Wobble artifacts are often largely imperceptible in single images, even in cases where the temporal artifacts in the video are quite dramatic. See the videos in the supplemental material (pausing at various points) for examples. About the only prior work that has come close to addressing rolling shutter wobble is [8]. This paper uses camera motion and context-preserving warps for video stabilization. Empirically, the authors noted that their algorithm tends to reducing rolling shutter wobble. However, the algorithm does not model the high-frequency temporal motion [10, 1] necessary for general purpose rolling shutter correction.

In this paper, we present an algorithm to remove rolling shutter wobble in video. In particular, we show how estimating the high-frequency jitter of the camera can be posed as a temporal super-resolution problem [3, 14]. The temporal low-frequency measurements (analogous of the low resolution pixels) are the motions of pixels from one frame to the next. These measurements are modeled as temporal integrals of the underlying high-frequency jitter of the camera. The estimated high-frequency motion of the camera is then used to re-render the video as though all the pixels in each frame were imaged at the same time.

We begin in Section 2.1 by deriving our algorithm for a high-frequency (e.g. per row) *translational* jitter model, analogous to the one in [6]. It is straight-forward to generalize this model to a per row *affine* model. See [2] for the details. In Section 2.2 we generalize the model to include *independently moving objects.* In particular, we model the motion of each pixel as the combination of a low-frequency independent motion and a high-frequency camera jitter.

Our algorithm has a single calibration parameter, the time between the capture of two subsequent rows as a fraction of the time between two subsequent frames. In Section 2.3 we investigate the calibration of this parameter. We first derive a closed-form expression relating the solution of the super-resolution constraints with the correct parameter to the solution with another setting (for the translational model). This result is important because it implies that the performance of our algorithm should be robust to the setting of the calibration parameter, a result which we empirically validate. Second, we present an auto-calibration algorithm

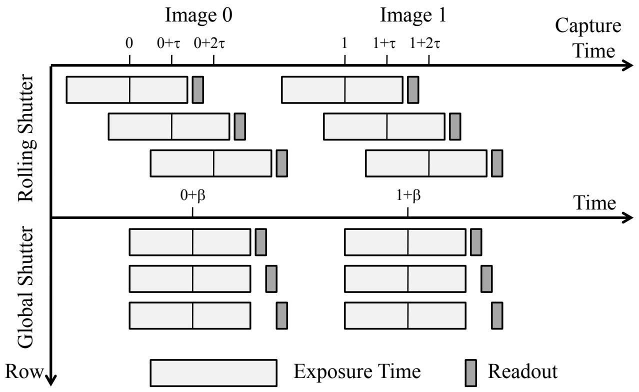


Figure 1. **Image Capture Model**: In a rolling shutter camera, each row is exposed and read out at a slightly later time than the previous row. We denote the difference in the capture times to be τ, where one time unit is the time between subsequent frames.

that can estimate the calibration parameter from a short segment of a video containing jitter.

2. Theory and Algorithms

Assume that the rolling shutter camera captures a video:

$$I_T^{\mathrm{RS}}(X, Y) \quad \text{for } T = 0, 1, 2, \ldots \tag{1}$$

Assume that the Y^{th} row in image $I_T^{\mathrm{RS}}(X, Y)$ is captured at time $T + \tau Y$, where we define the capture time of each row to be the mid-point of the exposure period for that row. See Figure 1 for an illustration. The non-zero exposure period means motion blur may be present. The shift over time in the mid-point of the exposure period still causes "rolling shutter" artifacts even in the presence of motion blur. Note that in this paper we do not address motion blur removal.

For now, we assume that τ has been calibrated and so is known. See Section 2.3 for a method to calibrate τ. We wish to correct the rolling shutter images $I_T^{\mathrm{RS}}(X, Y)$ to generate a sequence $I_T^{\mathrm{GS}}(X, Y)$ that might have been captured by a camera with a global shutter. We are free to choose the time $T + \beta$ that the global shutter image $I_T^{\mathrm{GS}}(X, Y)$ would have been captured. A natural choice is:

$$\beta \;=\; \tau \times (M - 1)/2 \tag{2}$$

because it minimizes the maximum correction and means that the center of the image will require the least correction. In this paper, we always use the value of β in Equation (2).

2.1. High-Frequency Translational Camera Jitter

We begin by describing our algorithm using a high-frequency (e.g. per row) translational model of the camera jitter. In this model, the motions of all the pixels in each row are assumed to be the same. The motion of each subsequent row can be different, however.

2.1.1 Motion Model

Denote the temporal trajectory of the projected location of a scene point $\mathbf{x}(t) = (x(t), y(t))$. We use lower case t, x, y to denote continuous variables and upper case T, X, Y to denote integer frame, column, and row numbers. Note that we model the continuous path of the point $\mathbf{x}(t)$ even through time periods that it is not imaged. If the camera is jittering, $\mathbf{x}(t)$ will vary rapidly between two subsequent frames T and $T + 1$. We assume that this high-frequency variation can be described using the following differential equation:

$$\frac{\mathrm{d}\,\mathbf{x}}{\mathrm{d}\,t} \;=\; \mathbf{m}^{\mathrm{hf}}(\mathbf{x}; \mathbf{p}(t)). \tag{3}$$

The parameters $\mathbf{p}(t)$ are a function of continuous time t. At any given time t, $\mathbf{m}^{\mathrm{hf}}(\mathbf{x}; \mathbf{p}(t))$ describes a low parametric spatial motion model. For example, $\mathbf{m}^{\mathrm{hf}}$ could be a translation. In this case, the parameter vector $\mathbf{p}(t) = (p_1(t), p_2(t))$ has two components, and:

$$\mathbf{m}^{\mathrm{hf}}(\mathbf{x}; \mathbf{p}(t)) \;=\; (p_1(t), p_2(t))\,. \tag{4}$$

In the remainder of this section, we use this translational model. See [2] for the derivation of our algorithm for a high-frequency (e.g. per row) affine model. In the translational model, at any given time t, all the points in the image are moving with the same motion. However, over the duration of a frame from T to $T + 1$, the translation may vary temporally. In the context of image deblurring with a global shutter camera, this translational model can result in arbitrarily complex blur kernels [6]. The blur kernels are the same at each pixel, however. In our case of a rolling shutter sequence, the low-frequency (frame-to-frame) motion of each row in the image can be different because each row is imaged at a different time. The temporally high-frequency translational model can result in complex, non-rigid image deformations; i.e. rolling shutter wobble.

Equation (3) defines a differential equation for $\mathbf{x}(t)$. To proceed, this equation must be solved. In the case of the translation, the continuous analytical solution is:

$$\mathbf{x}(t) \;=\; \mathbf{x}(t_0) + \int_{t_0}^{t} \mathbf{p}(s)\,\mathrm{d}s. \tag{5}$$

For more complicated motion models, deriving an analytic solution of Equation (3) may be impossible. An approximate or numerical solution can be used instead. See Section 2.2 for the approximation used in the case of independent motion and [2] for the affine case.

2.1.2 Measurement Constraints

We assume that measurements of the motion are available in the form of point correspondences. In this paper, all correspondences are obtained using the Black and Anandan optical flow algorithm [4]. Note that rolling shutter distortions

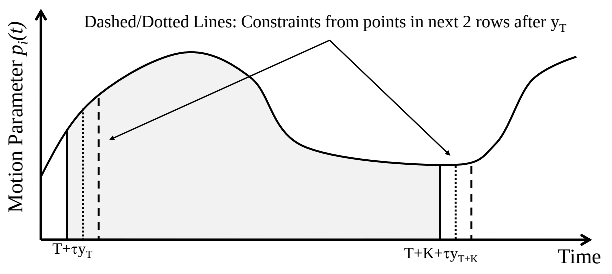

Dashed/Dotted Lines: Constraints from points in next 2 rows after y_T
Motion Parameter $p_i(t)$
$T+\tau y_T$
$T+K+\tau y_{T+K}$
Time

Figure 2. **Measurement Constraints:** Each constraint in Equation (7) specifies a known value for the integral of the unknown higher-resolution temporally varying motion parameters over a known interval. The constraints from points in nearby rows closely overlap each other, analogously to how sub-pixel shifts generate overlapping constraints in image super-resolution [3].

do not affect the extent to which brightness constancy holds. We subsample the flow fields, as described in detail in Section 3, to obtain a discrete set of correspondences. An alternative approach would be to use a feature detection and matching algorithms such as [9]. Note, however, that care should be taken as non-rigid image deformations do affect the values of most feature descriptors to some extent.

We assume each correspondence takes the form:

$$\mathrm{Corr}_i = (T_i, T_i + K_i, \mathbf{x}_{T_i}, \mathbf{x}_{T_i+K_i}) \,. \tag{6}$$

This correspondence means that a point at $\mathbf{x}_{T_i} = (x_{T_i}, y_{T_i})$ in image $I^{\mathrm{RS}}_{T_i}$ was matched, tracked, or flowed to the point $\mathbf{x}_{T_i+K_i} = (x_{T_i+K_i}, y_{T_i+K_i})$ in the second image $I^{\mathrm{RS}}_{T_i+K_i}$. The times T_i and $T_i + K_i$ are integers. Although in many cases, the first location $\mathbf{x}_{T_i}$ is integer valued, the second location $\mathbf{x}_{T_i+K_i}$ should be estimated with sub-pixel accuracy. For generality we denote both as real-valued.

Each correspondence can be substituted into Equation (5) to generate a measurement constraint:

$$\mathrm{MC}(\mathrm{Corr}_i) = \mathbf{x}_{T_i+K_i} - \mathbf{x}_{T_i} - \int_{T_i+\tau y_{T_i}}^{T_i+K_i+\tau y_{T_i+K_i}} \mathbf{p}(s)\,\mathrm{d}s \tag{7}$$

where ideally $\mathrm{MC}(\mathrm{Corr}_i) = 0$. Note that the integral is from the time that the point was imaged in the first image $T_i + \tau y_{T_i}$ to the time at which it was imaged in the second image $T_i + K_i + \tau y_{T_i+K_i}$; i.e. the length of the interval is not exactly K_i. Also note that the constraints in Equation (7) are temporal analogs of the constraints in image super-resolution [3]. Each constraint specifies a value for the integral of the unknown higher-resolution temporally varying motion parameters over a known interval. See Figure 2 for an illustration. The constraints from points in neighboring rows closely overlap each other, analogously to how sub-pixel shifts create overlapping constraints in [3].

One important difference is that the integral in Equation (7) is 1D (albeit of a 2D vector quantity), whereas in image super-resolution, it is a 2D integral (of 1-3 band images). Image super-resolution is known to be relatively poorly conditioned [3]. Obtaining resolution improvements beyond a factor of 4–6 or so is difficult. In [14], however, it was shown that 1D super-resolution problems are far better conditioned. Roughly speaking, the condition number in 1D is the square-root of the condition number in the corresponding 2D case. Consistent with this analysis, we encountered diminishing returns when attempting to enhance the temporal resolution by more than a factor of 30 or so.

2.1.3 Regularization and Optimization

We regularize the problem using a standard first order smoothness term that encourages the temporal derivative of the motion $\mathbf{p}$ to be small. We use L1 norms to measure errors in both the measurement constraints and regularization. We used the following global energy function:

$$\sum_{\mathrm{Corr}_i} |\mathrm{MC}(\mathrm{Corr}_i)| \;+\; \lambda \sum_{j=1,2} \int \left| \frac{\mathrm{d}\,p_j}{\mathrm{d}\,s} \right| \mathrm{d}s. \tag{8}$$

The measurement constraints are likely to contain a number of outliers, both due to independently moving objects and gross errors in the flow field. An L1 norm is therefore preferable to an L2 norm. We could also use an even more robust energy function, but such a choice would make the optimization more complex. We use an L1 norm rather than an L2 norm for the regularization term, as it is reasonable to expect the motion to be piecewise smooth, with discontinuities during rapid accelerations.

We represent the continuous motion parameters with a uniform sampling across time. The exact number of samples used in each experiment is reported in Section 3. As in [3], we use a piecewise constant interpolation of the samples when estimating the integral in the measurement constraints. With this representation, both the measurement constraints and the regularization term are linear in the unknown motion parameters. We solved the resulting convex L1 optimization using linear programming.

2.1.4 Correction Process

We wish to estimate the global shutter pixels $I^{\mathrm{GS}}_T(X, Y)$ using the rolling shutter pixels $I^{\mathrm{RS}}_T(x, y)$. We assume that X, Y, and T are known integer values, whereas x and y are unknown subpixel locations. Once we know x and y, we can (bicubically) interpolate the rolling shutter image. To estimate x and y, it helps to also estimate the time t at which this rolling shutter pixel was captured. Figure 3 contains an visualization of a 2D (y, t) slice through 3D (x, y, t) space. We project out the x variable and only show one pixel in each row of the image. Under the translational model, the motion of each pixel in a row is identical.

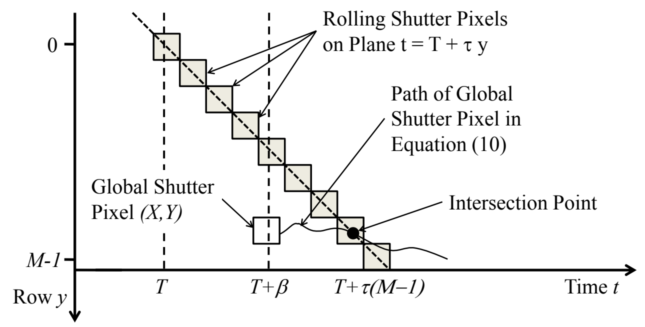


Figure 3. **Correction Process:** A 2D (y,t) slice through 3D (x,y,t) space. The rolling shutter pixels $I_T^{\mathrm{RS}}(x,y)$ lie on the plane $t = T+\tau y$. The global shutter pixel $I_T^{\mathrm{GS}}(X,Y)$ starts at 3D location $(X,Y,T+\beta)$ and moves along the path in Equation (10.) The correction process operates by first computing the intersection between the rolling shutter plane and the global shutter path. The rolling shutter image is then interpolated at this point.

The rolling shutter pixels $I_T^{\mathrm{RS}}(x,y)$ lie on the plane:

$$t = T + \tau y. \qquad (9)$$

Compensating for the estimated motion, the global shutter pixel $I_T^{\mathrm{GS}}(X,Y)$ starts at 3D location $(X,Y,T+\beta)$ and moves along the path:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} X \\ Y \end{pmatrix} + \begin{pmatrix} \int_{T+\beta}^{t} p_1(s)\,\mathrm{d}s \\ \int_{T+\beta}^{t} p_2(s)\,\mathrm{d}s \end{pmatrix}. \qquad (10)$$

The correction process begins by solving the pair of simultaneous Equations (9) and (10). Plugging Equation (9) into the y row of Equation (10) gives:

$$\frac{t-T}{\tau} \;=\; Y + \int_{T+\beta}^{t} p_2(s)\,\mathrm{d}s. \qquad (11)$$

The solution of this equation for t is independent of X and so this process only needs to be applied once per row. The solution can be obtained by stepping through the representation of the motion parameters $\mathbf{p}(t)$, considering each pair of samples in turn and approximating the integral in Equation (11). For the time interval between each pair of motion samples, Equation (11) is linear in the unknown t. It is therefore easy to check whether there is a solution in this interval. Note that, assuming the absolute value of the vertical motion $p_2(t)$ is bounded above by $\frac{1}{\tau} - \epsilon$ for some $\epsilon > 0$, the solution of Equation (11) is unique. A single pass can therefore be made through each neighboring pair of motion samples, with an early termination if a solution is found. If no solution is found, the pixel must have moved outside the image. Once the solution of Equation (11) has been computed for t, the corresponding (x,y) location in the rolling shutter image can be computed using Equation (10).

2.2. Adding Low-Frequency Independent Motion

The L1-based energy function in Equation (8) is relatively robust to outliers such as independently moving objects. The correction applied to independently moving objects, however, will ignore their independent motion. Independently moving objects may still have a residual shear. We now extend our algorithm to model independently moving objects and correct this residual shear. We use a low-frequency model of the independently moving objects for two reasons: (1) In most cases, independently moving objects undergo relatively slow acceleration. There are exceptions, of course, such as rotating helicopter blades. (2) Modeling independently moving objects with a high-frequency model would be extremely challenging and ambiguous.

We generalize the motion model in Equation (3) to:

$$\frac{\mathrm{d}\,\mathbf{x}}{\mathrm{d}\,t} \;=\; \mathbf{m}^{\mathrm{hf}}(\mathbf{x};\mathbf{p}(t)) + \mathbf{m}^{\mathrm{lf}}_{\lfloor t \rfloor}(\mathbf{x}) \qquad (12)$$

where $\mathbf{m}^{\mathrm{lf}}_0,\ \mathbf{m}^{\mathrm{lf}}_1,\ldots$ is a low-frequency motion (constant within each frame), but spatially the variation is dense. The low-frequency model $\mathbf{m}^{\mathrm{lf}}_{\lfloor t \rfloor}(\mathbf{x})$ can be thought of as a per-pixel flow field, where each pixel flows with a temporally constant velocity between each pair of frames.

The low-frequency term $\mathbf{m}^{\mathrm{lf}}_{\lfloor t \rfloor}(\mathbf{x})$ makes analytically solving Equation (12) hard, as the dependence on $\mathbf{x}$ is essentially arbitrary. To obtain an approximate solution, we assume that the spatial variation in $\mathbf{m}^{\mathrm{lf}}_{\lfloor t \rfloor}(\mathbf{x})$ is small and treat this term as a constant. Using the translational model of Equation (4) for the high-frequency term, the approximate solution of Equation (12) is:

$$\mathbf{x}(t) \;\approx\; \mathbf{x}(t_0) + \int_{t_0}^{t} \mathbf{p}(s)\,\mathrm{d}s + (t - t_o)\mathbf{m}^{\mathrm{lf}}_{\lfloor t \rfloor}(\mathbf{x}_{t_0}) \qquad (13)$$

which yields the measurement constraints:

$$\mathrm{MC}(\mathrm{Corr}_i) = \mathbf{x}_{T_i+K_i} - \mathbf{x}_{T_i} - \int_{T_i+\tau y_{T_i}}^{T_i+K_i+\tau y_{T_i+K_i}} \mathbf{p}(s)\,\mathrm{d}s$$
$$-(K_i + \tau y_{T_i+K_i} - \tau y_{T_i})\mathbf{m}^{\mathrm{lf}}_{T_i}(\mathbf{x}_{T_i}). \qquad (14)$$

We regularize the low-frequency model by adding the following two terms to the global energy function:

$$\gamma \sum_T \int \left\|\nabla \mathbf{m}^{\mathrm{lf}}_T(\mathbf{x})\right\|_1 \mathrm{d}\mathbf{x} + \epsilon \sum_T \int \left\|\mathbf{m}^{\mathrm{lf}}_T(\mathbf{x})\right\|_1 \mathrm{d}\mathbf{x}. \qquad (15)$$

The first term encourages the low-frequency model to vary smoothly across the image. We also spatially subsample $\mathbf{m}^{\mathrm{lf}}_T(\mathbf{x})$ to reduce the number of unknowns. See Section 3 for the details. The second term is needed to resolve an ambiguity between the low-frequency and high-frequency models. We favor the high-frequency model by adding a (very small) penalty to non-zero independent motion.

During the correction process, the path of the global shutter pixel in Equation (10) becomes:

$$\mathbf{x} \;=\; \mathbf{X} + \int_{T+\beta}^{t} \mathbf{p}(s)\,\mathrm{d}s + (t - T - \beta)\mathbf{m}^{\mathrm{lf}}_T(\mathbf{X}). \qquad (16)$$

Note that the time of intersection of this path with the plane of rolling shutter pixels in Equation (9) is no longer independent of X. The intersection therefore needs to be performed for each pixel, rather than just once for each row. Note that this process can be sped up by solving the intersection on a subsampled mesh and then upsampling.

2.3. Calibrating τ

The only image formation parameter in our model is τ, the time between the capture of neighboring rows (see Figure 1.) In some cases, it is possible to calibrate τ for a camera in the lab [5]. In many cases, however, all we have is a video obtained from an unknown source. Two key questions are: (1) how sensitive is our algorithm to an erroneous setting of τ, and (2) can we auto-calibrate τ? In Section 2.3.1, we address the first question by deriving a closed-form expression relating two solutions of the measurement constraints with different τs. This result indicates that the performance of our algorithm should be robust to the setting of τ, a result which we empirically validate in Section 3.4. In Section 2.3.2, we derive an auto-calibration algorithm to estimate τ from a short segment of the video.

2.3.1 Analyzing the Effect of Incorrect Calibration

We first introduce some notation. Suppose that the images have M rows. Denote the duty cycle $d = (M-1)\tau$. The camera is active capturing image $I_T^{\mathrm{RS}}(X,Y)$ between times T and $T+d$. Between $T+d$ and $T+1$, the camera is inactive in the sense that no new rows are imaged.

Now consider two solutions to the measurement constraints in Equation (7). Suppose the first solution uses the correct $\tau = \tau_1$, duty cycle $d_1 = (M-1)\tau_1$ and the second solution uses an incorrect $\tau = \tau_2$, duty cycle $d_2 = (M-1)\tau_2$. Let $r = d_1/d_2 = \tau_1/\tau_2$. Also, split the solutions into their active and inactive parts:

$$\mathbf{p}_i(t) = \begin{cases} \mathbf{p}_i^{\mathrm{act}}(t) & \text{if } t - \lfloor t \rfloor <= d_i \\ \mathbf{p}_i^{\mathrm{ina}}(t) & \text{if } t - \lfloor t \rfloor > d_i \end{cases} \qquad i = 1,2 \quad (17)$$

where $\mathbf{p}_i^{\mathrm{act}}(t)$ is the active part of the solution and $\mathbf{p}_i^{\mathrm{ina}}(t)$ is the inactive part.

Below we show that if all the correspondences have $K = 1$ and so take the form $(T, T+1, \mathbf{x}_T, \mathbf{x}_{T+1})$, and:

$$\mathbf{p}_2^{\mathrm{act}}(t) \approx r\mathbf{p}_1^{\mathrm{act}}(r(t - \lfloor t \rfloor) + \lfloor t \rfloor) + \mathbf{c}_{\lfloor t \rfloor} \qquad (18)$$

where:

$$\mathbf{c}_{\lfloor t \rfloor} = \frac{1}{d_2}\left[\int_{\lfloor t \rfloor + d_1}^{\lfloor t \rfloor + 1} \mathbf{p}_1^{\mathrm{ina}}(s)\,\mathrm{d}s - \int_{\lfloor t \rfloor + d_2}^{\lfloor t \rfloor + 1} \mathbf{p}_2^{\mathrm{ina}}(s)\,\mathrm{d}s\right] \qquad (19)$$

then the integrals in Equation (7) are the same:

$$\int_{T+\tau_2 y_T}^{T+1+\tau_2 y_{T+1}} \mathbf{p}_2(s)\,\mathrm{d}s = \int_{T+\tau_1 y_T}^{T+1+\tau_1 y_{T+1}} \mathbf{p}_1(s)\,\mathrm{d}s. \quad (20)$$

For a correspondence $(T, T+1, \mathbf{x}_T, \mathbf{x}_{T+1})$ with $K = 1$, the left hand side of Equation (20) is:

$$\int_{T+\tau_2 y_T}^{T+1+\tau_2 y_{T+1}} \mathbf{p}_2(s)\,\mathrm{d}s = \int_{T+\tau_2 y_T}^{T+d_2} \mathbf{p}_2^{\mathrm{act}}(s)\,\mathrm{d}s + \int_{T+d_2}^{T+1} \mathbf{p}_2^{\mathrm{ina}}(s)\,\mathrm{d}s + \int_{T+1}^{T+1+\tau_2 y_{T+1}} \mathbf{p}_2^{\mathrm{act}}(s)\,\mathrm{d}s. \quad (21)$$

Plugging Equation (18) into the first term on the right hand side gives:

$$\int_{T+\tau_2 y_T}^{T+d_2} r\mathbf{p}_1^{\mathrm{act}}(r(s - \lfloor s \rfloor) + \lfloor s \rfloor) + c_{\lfloor s \rfloor}\,\mathrm{d}s \qquad (22)$$

which after substituting $s' = r(s-T) + T$ simplifies to:

$$\int_{T+\tau_1 y_T}^{T+d_1} \mathbf{p}_1^{\mathrm{act}}(s')\,\mathrm{d}s' + c_T(d_2 - \tau_2 y_T). \qquad (23)$$

Similarly the third term simplifies to:

$$\int_{T+1}^{T+1+\tau_2 y_{T+1}} \mathbf{p}_1^{\mathrm{act}}(s')\,\mathrm{d}s' + c_T \tau_2 y_{T+1}. \qquad (24)$$

Assuming that $y_{T+1} \approx y_T$ and plugging these expressions into Equation (21) and substituting the expression for $c_{\lfloor t \rfloor}$ from Equation (19) yields Equation (20).

Our derivation makes one approximating assumption, that $y_T \approx y_{T+1}$. This assumption is reasonable because the vertical motion between two consecutive frames is generally only a small fraction of the frame.

Equation (18) provides a relationship between two solutions of the measurement constraints for different τ. Due to regularization and discretization, the final solution obtained by our algorithm will not exactly match Equation (18). It can be expected to hold approximately, however.

What does Equation (18) mean in terms of the final correction applied? First, note that only the active part of the solution is used in the correction process. See Figure 3. Second, note that if $\mathbf{c}_{\lfloor t \rfloor} = \mathbf{0}$ then Equation (18) would mean that exactly the same correction is applied for the two different values of τ. The proof of this fact follows the same argument as above. The difference in the two corrections is due to $\mathbf{c}_{\lfloor t \rfloor}$, a constant motion for each frame. The two corrections will therefore approximately differ by a global affine warp. In general, $\mathbf{c}_{\lfloor t \rfloor}$ will vary from frame to frame, as $\mathbf{c}_{\lfloor t \rfloor}$ is related to the motion in the inactive period.

In summary, with a slightly incorrect value of τ theory shows that the final corrections with approximately differ by a slightly different affine warp for each frame. Although estimating τ wrongly may add a little jitter to the output, our algorithm can be expected to be robust in the sense that there is little danger of gross artifacts being added.

2.3.2 An Auto-Calibration Algorithm

The analysis above suggests an algorithm to auto-calibrate τ from a short segment of the video containing jitter. We first perform the correction for a sampling of different values of $\tau \in [0, \frac{1}{M-1}]$. For each solution, we attempt to detect the small residual affine jitter from frame to frame that the above analysis predicts. In particular, we compute optical flow across each corrected result. We then compute a measure of how "translational" the motion is.

For each flow field between a pair of subsequent images, we consider a number of patches, compute the average flow, and then measure the median deviation of each pixel from this median translational motion. We then compute the median value of this measure across a sampling of patches in each flow field. We compute the mean value of this measure across the short segment of the video. Finally, we plot the measure across τ, smooth the result slightly, and then choose τ to take the minimum value. Note that although the analysis in Section 2.3.1 assumes a translation jitter model and makes several approximations, the auto-calibration is a reasonable approach in a much wider setting. Also note that, although this algorithm amounts to a brute force search, the 1D search range can be sampled sparsely and the algorithm only needs to be run on a short segment of the input video, so long as it contains some wobble.

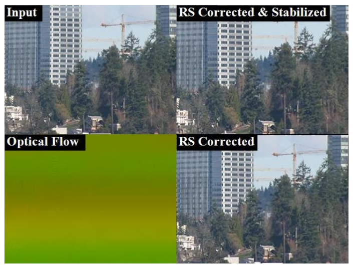
3. Experimental Results

To avoid the size of the optimization growing arbitrarily with the length of the video, we solved Equation (8) for a fixed size window that is slid one frame at a time through the video. Empirically we found a significant improvement in performance up to a window size of around 7-11 frames. All the results in this paper use a window size of 9 frames.

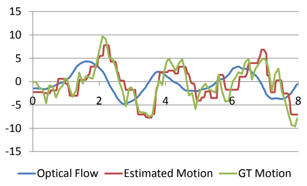
We obtained correspondences in the form of Equation (6) by sub-sampling optical flow fields computed using the Black and Anandan algorithm [4]. In the independent motion case, we sample the flow every row and every 10th column, excluding flows within 6 pixels of the edge of the image to avoid boundary errors in the flow field. In the translational case, the correction is constant along each row. We therefore use a single correspondence per row, obtained by median filtering the correspondences along each row. We experimented with $K > 1$ but found only a small gain. The results in this paper all use motion computed with $K = 1$.

We experimented with different sample rates for the motion parameters $\mathbf{p}$. We found diminishing returns beyond around 25-35 samples per frame. All results in this paper use 30 samples of $\mathbf{p}$ (60 unknowns) per frame. We subsample $\mathbf{m}_i^{\mathrm{lf}}(\mathbf{x})$ spatially every 10 rows and every 10 columns.

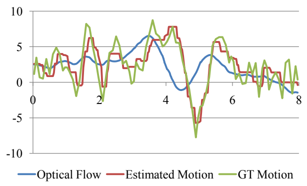
With the translational model we use the regularization weight $\lambda = 300$. The independent motion regularization weights are $\gamma = 10$ and $\epsilon = 1$. Most of our test sequences



(a) Example Frame From Synthetically Generated Movie



(b) Motion in x Direction



(c) Motion in y Direction

Figure 4. **An Illustrative Example**: (a) One frame from the video `city.mp4`. Our algorithm first computes the optical flow (bottom left). We then compute a corrected video without the non-rigid wobble artifacts, but which still contains *global* jitter (bottom right). We then stabilize the result (top right.) (b) and (c) A comparison of the low-frequency optical flow (blue) with the estimated (red) and ground-truth (green) high-frequency motion.

were downloaded from the web and so we do not know the camera details. For all video sequences, we estimated τ using the algorithm in Section 2.3.2. We generally found that $\tau \in [0.75/(M-1), 0.95/(M-1)]$. Note that Gunnar Thalin has calibrated a number of cameras in a laboratory setting and published the results online [15].

3.1. An Illustrative Synthetic Example

We begin with a synthetic example to illustrate the steps in our algorithm. In Figure 4(a) we include one frame from the video `city.mp4` included in the supplemental material. In the top left we show the input rolling shutter video which was generated synthetically using the translation model in Equation (3). In the bottom left we include the optical flow, where the horizontal flow is coded in the red channel and the vertical flow is coded in the green channel. In the bottom right, we include the corrected video, which still contains *global* high-frequency jitter. The non-rigid wobble has been removed, however. We include a stabilized version of the corrected output in the top right. We stabilize the videos by simply low-pass filtering the motion of the center of the frame and then applying a global correction for each frame. More sophisticated algorithms such as [8] could now be applied to the corrected video because all the pixels in each frame have been re-rendered as though they were captured at the same time.

In Figures 4(b) and (c) we compare the input optical flow (median filtered across each row), the estimated high-frequency motion, and the ground-truth high-frequency motion. We plot the value of the flow/motion on the y-axis. On the x-axis we plot time, which in the case of the optical flow

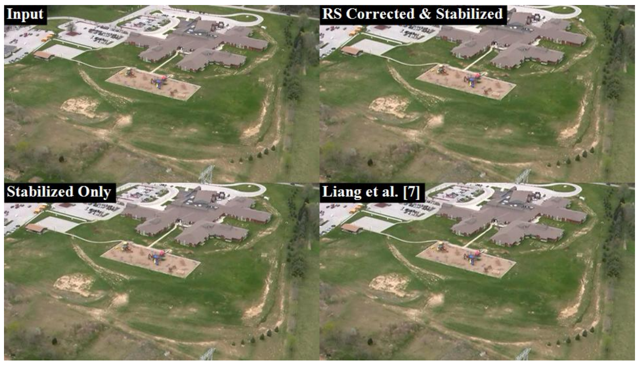


Figure 5. **A Qualitative Comparison**: One frame from the video skool.mp4. We compare the output of our algorithm (top right) with the result of naive stabilization (bottom left) and the result obtained with the algorithm described in [7] (bottom right).

corresponds to the row in the video that the flow was measured at. Note how the optical flow is relatively smoothly varying across time, whereas both the ground-truth and estimated motions are higher frequency. There is also a phase shift between the optical flow and the real camera motion.

We also include a video shear.mp4 comparing the input, output, and ground-truth for a similar synthetic sequence generated with constant motion. This video confirms that our algorithm handles a simple shear correctly.

3.2. Qualitative Comparisons

Our main evaluation consists of a set of qualitative comparisons. In Figure 5 we include one frame from the video skool.mp4. In the top left we include the input. In the top right we include the stabilized output of our algorithm. In the bottom left we include the result of stabilization without correcting the rolling shutter distortions.

We implemented the algorithm in [7] and the morphing algorithm in [5]. Empirically, we found the algorithm in [7] to outperform the one in [5]. For lack of space, we only present the results for the algorithm in [7] in the bottom right of Figure 5 and the video. When implementing [7] we used the same Black and Anandan flow [4] used by our algorithm, rather than the block matching described in [7]. We also used the median filtered flow for every row, as in our algorithm, rather that the four samples in [7]. These changes should only improve the algorithm in [7] and make the comparison fairer. The algorithm in [7] does not perform any high-frequency analysis or super-resolution. Instead it performs an interpolation of the motion. While [7] corrects some artifacts, it does not remove all the wobble.

We also include several other videos in the supplemental material. The video vegas1.mp4 is another aerial video, which shows the robustness of our algorithm to low light conditions, saturation, and motion blur (which we do not attempt to remove.) The video bike.mp4 is robust to very noisy input, where the computed flow is very noisy. The video reverse.mp4 illustrates that our algorithm performs reasonably even in the presence of some rotational motion. Finally, race.mp4 contain footage from a car involved in a high-speed race. Most of the video is well corrected. It does, however, illustrate one failure case of our algorithm. As the car goes over the rumble strips, the optical flow algorithm fails completely due to the large induced motion. These errors lead to a couple of "bumps" in the output video. The output video is still a dramatic improvement over the input, however.

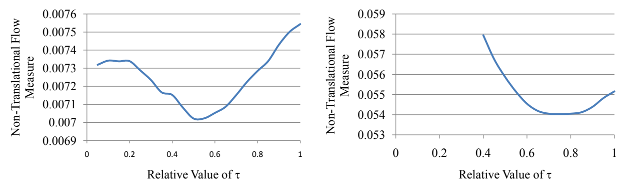
3.3. Independent Motion

We first illustrate the independent motion extension to our algorithm on a synthetic dataset in order to present a comparison with ground-truth. In balloon.mp4 we compare the results obtained using just the translational model (bottom right) with the results obtained using the independent motion model in addition to the translational model (top right). In the bottom left we include results using the morphing algorithm of [5] which is specifically designed to handle independent motion. As can be seen, however, the morphing algorithm of [5] cannot handle high-frequency jitter. Note that even just the translational model is robust to the independent motion, through the median filtering of the flows and the L1 norms. Finally, note that the residual shear on the balloon is largely imperceptible in balloon.mp4. When toggled with the ground-truth in toggle.mp4, the residual shear becomes more apparent.

In checker.mp4 we include an example of a rotating checkerboard, similar to an example presented in [5]. Our example also contains high-frequency jitter not present in the example in [5]. While the motion in checker.mp4 could be modeled with our affine model, rotational motion is a useful tool when evaluating independent motion modeling because errors make straight lines appear curved. As can be seen in checker.mp4, with the addition of the independent motion model, the curvature disappears.

3.4. Calibration

We first re-generated city.mp4 using $\tau = \frac{1}{2}\frac{1}{M-1}$; i.e. half of the maximum value. In Figure 6(a) we present the results of our auto-calibration algorithm. These show a clear minimum close to the ground-truth value of 0.5. In Figure 6(b) we present calibration results for the skool.mp4 video. We downloaded this video from the web and so do not know the ground-truth value of τ. In skool_cal.mp4 we present a qualitative comparison of the affect of varying the relative value (i.e. multiplied by $M-1$) of τ. These qualitative results confirm two things. First, the calibrated relative value of $\tau = 0.75$ does appear to be reasonable. Second, our algorithm is relatively insensitive to the exact choice of τ, validating Section 2.3.1.



(a) Calibration on city.mp4 (b) Calibration on heli.mp4

Figure 6. **Quantitative Calibration Results:** (a) Auto-calibration results on a version of `city.mp4` where the ground-truth relative value of τ is 0.5. (b) Auto-calibration results on `skool.mp4`. In `skool_cal.mp4` we present a comparison of $\tau = 0.25$, 0.5, 0.75, and 1.0. This video confirms that the calibration result of 0.75 is reasonable and illustrates the robustness of our algorithm.

3.5. Timing Results

For our timing results, we have made no attempt to implement our algorithm efficiently and use the relatively slow Black and Anandan algorithm [4]. We timed our algorithm on the 30 frame, 320×240 pixel video `city.mp4`, running our algorithms on a 2.0Ghz Dual-Core HP nc8430 laptop. Computing the flow and extracting the correspondences took 7.6 seconds per frame. Solving the super-resolution problem took 2.1 seconds per frame for the translational model and 102.4 seconds per frame for the independent motion model. Correcting the distortions and stabilizing the video took 0.2 seconds per frame. One way to speed up our algorithm is to run the flow and super-resolution on a downsampled video. The computed high-frequency motion can then be scaled up and applied to the original video. In `timing.mp4` we compare results obtained on `city.mp4` at the full resolution with those on a half-size video. There is little difference, indicating that speed-ups are possible.

4. Conclusion

We have presented an algorithm to remove rolling shutter wobble. Our algorithm uses a form of temporal super-resolution to infer the high-frequency motion of the camera from low-frequency optical flow. We extended our algorithm to use an affine motion model and to model low-frequency independent motion. We showed both analytically and empirically that our algorithm is robust to the setting of the key calibration parameter τ. We also presented an auto-calibration algorithm that can estimate this parameter from a short segment of the video containing jitter.

One failure mode of our algorithm occurs when the motion is so great that the optical flow algorithm completely fails; e.g. `race.mp4` in Section 3.2. One possible solution is to fall back on sparse feature matching [9] in such cases. The reduced density of correspondences will result in less accurate, but hopefully more robust results. Secondly, our model is currently unable to model large parallax between foreground and background objects. Finally, on very close inspection, some residual wobble can be seen in the output, caused by the inherent limitations of the super-resolution process [3, 14]. Novel priors or longer-range correspondences ($K > 1$) could possibly reduce the residual wobble.

One possible future direction is to investigate the choice of the error functions and optimization algorithm, both to improve quality and speed. Another possibility is to explore the direct estimation of the motion model parameters.

Acknowledgements

Simon Baker would like to thank Iain Matthews for suggesting working on this problem. We thank Michael Black for providing his implementation of the Black and Anandan algorithm [4]. Finally, thanks to Russ Andersson of Andersson Technologies LLC (http://www.ssontech.com/) for providing the `skool.mp4` sequence.

References

[1] O. Ait-Aider, A. Bartoli, and N. Andreff. Kinematics from lines in a single rolling shutter image. In *CVPR*, 2007.

[2] S. Baker, E. Bennett, S. Kang, and S. Szeliski. Removing rolling shutter wobble. Technical Report MSR-TR-2010-28, Microsoft Research, 2010.

[3] S. Baker and T. Kanade. Limits on super-resolution and how to break them. *IEEE PAMI*, 24(9):1167–1183, 2002.

[4] M. J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *CVIU*, 63(1):75–104, 1996.

[5] D. Bradley, B. Atcheson, I. Ihrke, and W. Heidrich. Synchronization and rolling shutter compensation for consumer video camera arrays. In *Proceedings of International Workshop on Projector-Camera Systems (PROCAMS)*, 2009.

[6] R. Fergus, B. Singh, A. Hertzmann, S. Roweis, and W. Freeman. Removing camera shake from a single photograph. *ACM Transactions on Graphics*, 25(3):787–794, 2006.

[7] C.-K. Liang, L.-W. Chang, and H. Chen. Analysis and compensation of rolling shutter effect. *IEEE TIP*, 17(8):1323–1330, 2008.

[8] F. Liu, M. Gleicher, H. Jin, and A. Agarwala. Context-preserving warps for 3D video stabilization. In *SIGGRAPH*, 2009.

[9] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[10] M. Meingast, C. Geyer, and S. Sastry. Geometric models of rolling-shutter cameras. In *Proc. of the Intl. Wkshp. on Omni. Vision, Cam. Networks, and Non-Classical Cams.*, 2005.

[11] S. Nicklin, R. Fisher, and R. Middleton. Rolling shutter image compensation. In *Proceedings of RoboCup*, 2006.

[12] Pure Digital Technologies, LLC. Flip video camcorder. http://www.theflip.com/.

[13] Red.com, Inc. Red digital camera. http://www.red.com/.

[14] E. Shechtman, Y. Caspi, and M. Irani. Space-time super-resolution. *IEEE PAMI*, 27(4):531–545, 2005.

[15] G. Thalin. Deshaker rolling shutter settings. http://www.guthspot.se/video/deshaker.htm#rolling shutter setting.