Removing Rolling Shutter Wobble

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

March 2010

Technical Report
MSR-TR-2010-28

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 high-frequency estimated 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.

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
http://www.research.microsoft.com

1 Introduction

Whereas historically video cameras and camcorders primarily used CCD sensors, 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 effective capture time of each row is slightly after that of the previous row.

Rolling shutter cameras suffer from three main artifacts [16]: (1) skew, (2) partial exposure, and (3) wobble. Skewing occurs when the camera undergoes a constant (or smoothly varying) motion. Skewing 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. This effect can result in images that are darker in some regions and lighter in others [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. Figure 1(a) contains one frame from a video with wobble artifacts. The straight lines on the building have become curved. Note, however, that wobble artifacts are far more apparent in videos than they are in any single frame. See the videos in the supplementary material 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.

(a) Rolling Shutter Input

(b) Corrected Output

Figure 1: **Example**: (a) One frame from a video with rolling shutter wobble artifacts. (b) The output of our algorithm. Notice how the curved lines on the building have become far more straight. Note, however, that rolling shutter wobble artifacts are far more apparent in the video than they are in any single frame. See the videos in the supplementary material for examples.

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]. In Section 2.2 we show how to generalize this model to a high-frequency *affine* model. In Section 2.3 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, namely, the time between the capture of two subsequent rows as a fraction of the time between two subsequent frames. This parameter is a measure of how severe the rolling shutter is. When the parameter is zero, the camera has a global shutter. As the parameter increases, rolling shutter artifacts become more pronounced.

In Section 2.4 we investigate the calibration of this parameter. We first derive a closed-form expression relating the solution of the temporal super-resolution constraints with the correct parameter to the solution with another setting (for the translational model). This result is important because it indicates 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 that can estimate the calibration parameter from a short segment of a video containing some jitter.

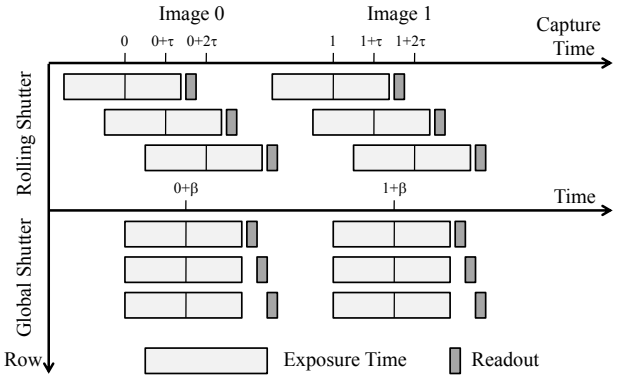


Figure 2: **Image Capture Model**: In a rolling shutter camera, each row is exposed and read out at a slightly later time than the previous row. We denote the difference in the capture times to be τ, where one time unit is the time between subsequent frames.

2 Theory and Algorithms

Denote the rolling shutter 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 2 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 [6].

For now, we assume that τ is known or has been calibrated. See [5] for an algorithm to calibrate a camera in the lab. In Section 2.4 we present a method to perform auto-calibration for a video obtained from a camera that is no longer available for calibration. 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. If the number of rows in the images is M, a natural choice for β is:

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

because it tends to minimize 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). Note that β is not a calibration parameter, but can be specified arbitrarily.

2.1 High-Frequency Translational Jitter

We first consider a high-frequency translational model of the camera jitter. By high-frequency, we mean that each row in the image could potentially have a different translation. The motion of all the pixels in each row are assumed to be the same, however. The actual model is a continuous function of time; at any given continuous time t, we model the instantaneous translational motion. We only discretize the model into a finite sampling of parameters to perform the optimization. Our translational jitter model is analogous to the one used in [6] for motion blur. Empirically we found it to be a good approximation for many videos.

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 the capture of two subsequent frames T and $T+1$. We assume that this high-frequency variation can be described by 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 Section 2.2 for the derivation of our algorithm

for the corresponding affine model. In the translational model, 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 arbitrarily. In the context of image deblurring with a global shutter camera [6], this translational model can result in arbitrarily complex blur kernels. 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. Although a translation model may seem too simple, the fact that it is high-frequency allows it to explain many non-rigid image deformations such as those in Figure 1 that are perceived as wobble in rolling shutter video.

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

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

Deriving an analytic solution of Equation (3) for more complicated motion models may be impossible. In such case, an approximate or numerical solution must be used instead. See Section 2.2 (affine) and Section 2.3 (independent motion) for two examples.

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 do not affect the extent to which brightness constancy holds. We subsample the optical 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 algorithm such as [9]. Note, however, that care should be taken as non-rigid image deformations do affect the values of most feature descriptors.

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 expression means that a point $\mathbf{x}_{T_i} = (x_{T_i}, y_{T_i})$ in image $I^{\mathrm{RS}}_{T_i}$ was matched, tracked, or flowed to the corresponding 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 consistency and generality we denote both as real-valued.

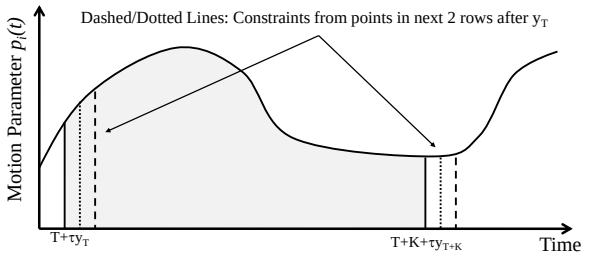


Figure 3: **Measurement Constraints:** Each constraint in Equation (7) specifies a known value for the integral of the unknown higher-resolution temporally varying motion parameters over a known interval. The constraints from points in nearby rows closely overlap each other, analogously to how sub-pixel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

Each correspondence generates a measurement constraint by substituting into Equation (5):

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

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 3 for an illustration. The constraints from points in neighboring rows closely overlap each other, analogously to how sub-pixel shifts create overlapping constraints in image super-resolution [3].

One important difference between our problem and image super-resolution is that the integral in Equation (7) is 1D (albeit of a 2D vector quantity), whereas in image

super-resolution, the constraints are 2D area integrals (of 1-3 band images). Image super-resolution is known to be relatively poorly conditioned [3]. Obtaining resolution enhancement beyond a factor of 4–6 or so is difficult, even in ideal conditional. 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 theoretical analysis, empirically we only encountered diminishing returns when attempting to enhance the temporal resolution by a factor of more than 30 or so.

2.1.3 Regularization and Optimization

We regularize the measurement constraints using a 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. In particular, we used the following global energy function:

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

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 also 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 (near) discontinuities during very rapid accelerations.

We represent the continuous motion $\mathbf{p}$ with a uniform sampling across time into a finite number of parameters. We typically used between 25 and 35 samples per image (around 1 sample every 10 rows in a 320×240 image or 1 sample every 20 rows for a 640×480 image). As mentioned above, higher sampling rates yielded diminishing returns and simply require more computation. 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.

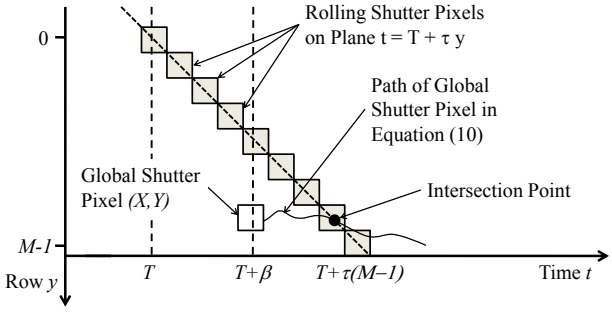


Figure 4: **Correction Process:** A 2D (y, t) slice through 3D (x, y, t) space. The rolling shutter pixels $I_T^{\text{RS}}(x, y)$ lie on the plane $t = T + \tau y$. The global shutter pixel $I_T^{\text{GS}}(X, Y)$ starts at 3D location $(X, Y, T + \beta)$ and moves along the path in Equation (10.) The 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.

2.1.4 Correction Process

We wish to estimate the global shutter pixels $I_T^{\text{GS}}(X, Y)$ using the rolling shutter pixels $I_T^{\text{RS}}(x, y)$. 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 4 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.

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

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

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

moves along the path:

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

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:

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

The solution of Equation (11) for t is independent of X. For the translational model, the correction:

$$\begin{pmatrix} \int_{T+\beta}^{t} p_1(s)\,\mathrm{d}s \\ \int_{T+\beta}^{t} p_2(s)\,\mathrm{d}s \end{pmatrix} \quad (12)$$

for each pixel in a row is the same. Equation (11) therefore only needs to be solved once per row. The solution of Equation (11) can be obtained by stepping through the discrete 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 not too large (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 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 correction in Equation (12) can be computed and then applied to each pixel in the row using Equation (10).

2.2 Affine Model of Camera Jitter

We now consider a high-frequency affine model of the camera jitter. It simplifies the equations to use two sets of parameters, the 2D translational motion vector $\mathbf{p}(t) = (p_1(t), p_2(t))$ and the 2×2 matrix:

$$\mathbf{q}(t) = \begin{pmatrix} q_1(t) & q_2(t) \\ q_3(t) & q_4(t) \end{pmatrix}. \quad (13)$$

The six affine parameters at each time t can be concatenated into a single vector $(p_1, p_2, q_1, q_2, q_3, q_4)$ if so desired. In this paper, we just keep two sets of parameters, the vector $\mathbf{p}(t)$ and the matrix $\mathbf{q}(t)$. We then define the high-frequency affine jitter model to be:

$$\frac{\mathrm{d}\,\mathbf{x}}{\mathrm{d}\,t} = \mathbf{m}^{\mathrm{hf}}(\mathbf{x}; \mathbf{p}(t); \mathbf{q}(t)) = \mathbf{p}(t) + \mathbf{x}\mathbf{q}(t). \quad (14)$$

The solution of Equation (14) can be approximated:

$$\mathbf{x}(t) \approx \mathbf{x}(t_0) + \int_{t_0}^{t} \left[\mathbf{p}(s) + \mathbf{x}(t_0)\mathbf{q}(s)\right]. \quad (15)$$

Equation (15) is approximate in the following way. First note that Equation (14) is a differential definition. The parameters $\mathbf{p} = \mathbf{0}$, $\mathbf{q} = \mathbf{0}$ correspond to the identity transformation since $\frac{\mathrm{d}\,\mathbf{x}}{\mathrm{d}\,t} = 0$ corresponds to the identity. The corresponding finite difference affine transform to the one in Equation (14) would use the matrix:

$$\begin{pmatrix} 1 + q_1(t) & q_2(t) \\ q_3(t) & 1 + q_4(t) \end{pmatrix}; \quad (16)$$

i.e. the differential definition models the change that must be applied in addition to the identity transformation. The parameters of the composition of two affine transforms parameterized using Equation (16) is equal to the sum of the parameters *neglecting second order terms.* Integrating the differential definition in Equation (14) therefore corresponds to integrating the parameters neglecting second order terms.

We validated the approximation in Equation (15) empirically. With reasonable values for $\mathbf{p}(t)$ and $\mathbf{q}(t)$, we found that a difference between a correctly warped image and one approximated using Equation (15) only begins to appear visually after accumulating the warps for over 15 frames ($t - t_0 > 15$). In our algorithms, the approximation only needs to hold for the duration of the measurement constraints K. In all our experiments $K = 1$ and in most reasonable scenarios $K << 15$.

Given Equation (15), the measurement constraints in Equation (6) become: $\mathrm{MC}(\mathrm{Corr}_i) =$

$$\mathbf{x}_{T+K} - \mathbf{x}_T - \int_{T+\tau y_T}^{T+K+\tau y_{T+K}} \left[\mathbf{p}(s) + \mathbf{x}_T\mathbf{q}(s)\right]\,\mathrm{d}s. \quad (17)$$

We add a regularization term for $\mathbf{q}$:

$$\delta \sum_{j=1}^{4} \int \left| \frac{\mathrm{d}\, q_j}{\mathrm{d}\, s} \right| \mathrm{d}s \tag{18}$$

to the global energy function in Equation (8). The path of the global shutter pixel in the correction process changes from Equation (10) to:

$$\mathbf{x} \;=\; \mathbf{X} + \int_{T+\beta}^{t} \left[\mathbf{p}(s) + \mathbf{X}\mathbf{q}(s) \right]. \tag{19}$$

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. This process can be sped up if so desired, albeit introducing a small approximation, by solving the intersection on a subsampled mesh and then upsampling.

2.3 Low-Frequency Independent Motion

The L1-based energy function in Equation (8) is relatively robust to outliers. Empirically, we find that the algorithms in Sections 2.1 and 2.2 lock onto the dominant motion, even in the presence of fairly large independently moving objects, and whether the global camera motion is jittery, smooth, or zero.

The correction applied to independently moving objects ignores their independent motion, however. Independently moving objects may still have residual deformations that are uncompensated. We now extend our translational algorithm in Section 2.1 to explicitly model independently moving objects and correct for their independent motion. A similar extension could also be derived for the affine algorithm in Section 2.2.

We use a low-frequency model of the independently moving objects for two reasons. First, in most cases, independently moving objects undergo relatively slow acceleration. There are exceptions, of course, such as rotating helicopter blades. However, common cases such as people moving and cars passing are relatively low frequency. Second, modeling independently moving objects with a high-frequency model would be extremely challenging and ambiguous. Such a model would require a large number of unknowns for each pixel in the video. Sampling at roughly the same frequency as the translational model, say, 30 samples per frame, would require 60 unknowns for each pixel in the video. Such a formulation of the problem, while conceivable, would require very aggressive regularization and would likely be very sensitive to noise in the input flow fields.

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

$$\frac{\mathrm{d}\,\mathbf{x}}{\mathrm{d}\,t} \;=\; \mathbf{m}^{\mathrm{hf}}(\mathbf{x}; \mathbf{p}(t)) + \mathbf{m}^{\mathrm{lf}}_{\lfloor t \rfloor}(\mathbf{x}), \tag{20}$$

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

The low-frequency term $\mathbf{m}^{\mathrm{lf}}_{\lfloor t \rfloor}(\mathbf{x})$ makes analytically solving Equation (20) 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 (20) is:

$$\mathbf{x}(t) \;\approx\; \mathbf{x}(t_0) + \int_{t_0}^{t} \mathbf{p}(s)\, \mathrm{d}s + (t - t_o)\mathbf{m}^{\mathrm{lf}}_{\lfloor t \rfloor}(\mathbf{x}_{t_0}) \tag{21}$$

which yields the measurement constraints: $\mathrm{MC}(\mathrm{Corr}_i)$

$$= \mathbf{x}_{T_i+K_i} - \mathbf{x}_{T_i} - \int_{T_i+\tau y_{T_i}}^{T_i+K_i+\tau y_{T_i+K_i}} \mathbf{p}(s)\, \mathrm{d}s \tag{22}$$

$$-(K_i + \tau y_{T_i+K_i} - \tau y_{T_i}) \mathbf{m}^{\mathrm{lf}}_{T_i}(\mathbf{x}_{T_i}). \tag{23}$$

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

$$\gamma \sum_T \int \left\| \nabla \mathbf{m}^{\mathrm{lf}}_T(\mathbf{x}) \right\|_1 \mathrm{d}\mathbf{x} + \epsilon \sum_T \int \left\| \mathbf{m}^{\mathrm{lf}}_T(\mathbf{x}) \right\|_1 \mathrm{d}\mathbf{x}. \tag{24}$$

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. 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:

$$\mathbf{x} \;=\; \mathbf{X} + \int_{T+\beta}^{t} \mathbf{p}(s)\,\mathrm{d}s + (t - T - \beta)\mathbf{m}_T^{\mathrm{lf}}(\mathbf{X}). \quad (25)$$

As is the case for the affine model in Section 2.2, the time of intersection of this path with the plane of rolling shutter pixels in Equation (9) depends on X. The intersection needs to be performed independently for each pixel, rather than just once for each row. Again, note that this process can be sped up, by solving for the correction on a subsampled mesh and then upsampling.

2.4 Calibrating τ

The only image formation parameter in our model is τ, the time between the capture of neighboring rows (see Figure 2.) 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 2.4.1, we address the first question by deriving an expression relating two solutions of the measurement constraints with different values of τ. This result indicates that the performance of our algorithm should be robust to the exact setting of τ, a result which we empirically validate in Section 3.5. In Section 2.4.2, we derive an auto-calibration algorithm to estimate τ from a short segment of jittery video.

2.4.1 Analyzing the Effect of Incorrect Calibration

Denote the duty cycle $d = (M-1)\tau$, where M is the number of rows in the video. The camera is active capturing image $I_T^{\mathrm{RS}}(X, Y)$ between time T and time $T + d$. Between time $T+d$ and time $T+1$, the camera is inactive in the sense that no new rows are imaged. See Figure 5 for a visualization.

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

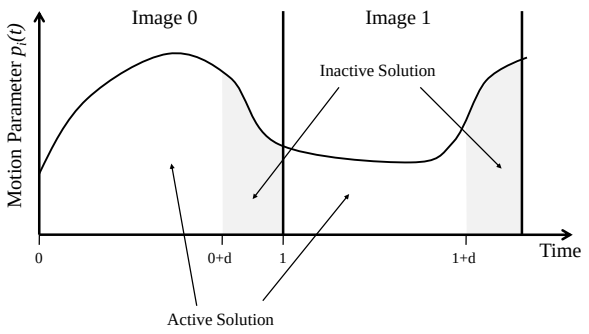


Figure 5: **Analysis of Calibration:** In general, rolling shutter cameras are not capturing rows of the image at all times. Typically, the camera is active, capturing images over the duty cycle from time T to time $T + d$ where $d = (M-1)\tau$ and M is the number of rows in the image. In Equation (27) we derive an expression relating two solutions of the measurement constraints during the active period for two different settings of τ.

solutions into their active and inactive parts:

$$\mathbf{p}_i(t) = \begin{cases} \mathbf{p}_i^{\mathrm{act}}(t) & \text{if } t - \lfloor t \rfloor <= d_i \\ \mathbf{p}_i^{\mathrm{ina}}(t) & \text{if } t - \lfloor t \rfloor > d_i \end{cases} \qquad i = 1, 2 \quad (26)$$

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:

$$\mathbf{p}_2^{\mathrm{act}}(t) \approx r\mathbf{p}_1^{\mathrm{act}}(r(t - \lfloor t \rfloor) + \lfloor t \rfloor) + \mathbf{c}_{\lfloor t \rfloor} \quad (27)$$

where:

$$\mathbf{c}_{\lfloor t \rfloor} \;=\; \frac{1}{d_2}\left[\int_{\lfloor t \rfloor + d_1}^{\lfloor t \rfloor + 1} \mathbf{p}_1^{\mathrm{ina}}(s)\,\mathrm{d}s - \int_{\lfloor t \rfloor + d_2}^{\lfloor t \rfloor + 1} \mathbf{p}_2^{\mathrm{ina}}(s)\,\mathrm{d}s\right] \quad (28)$$

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

$$\int_{T+\tau_2 y_T}^{T+1+\tau_2 y_{T+1}} \mathbf{p}_2(s)\,\mathrm{d}s \;=\; \int_{T+\tau_1 y_T}^{T+1+\tau_1 y_{T+1}} \mathbf{p}_1(s)\,\mathrm{d}s. \quad (29)$$

For a correspondence $(T, T+1, \mathbf{x}_T, \mathbf{x}_{T+1})$ with $K = 1$, the left hand side of Equation (29) is:

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

$$\int_{T+d_2}^{T+1} \mathbf{p}_2^{\text{ina}}(s)\,\mathrm{d}s + \int_{T+1}^{T+1+\tau_2 y_{T+1}} \mathbf{p}_2^{\text{act}}(s)\,\mathrm{d}s. \quad (30)$$

Plugging Equation (27) into the first term on the right hand side of Equation (30) gives:

$$\int_{T+\tau_2 y_T}^{T+d_2} r\mathbf{p}_1^{\text{act}}(r(s-\lfloor s \rfloor)+\lfloor s \rfloor) + c_{\lfloor s \rfloor}\,\mathrm{d}s \quad (31)$$

which after substituting $s' = r(s-T)+T$ and $\mathrm{d}s' = r\mathrm{d}s$ simplifies to:

$$\int_{T+\tau_1 y_T}^{T+d_1} \mathbf{p}_1^{\text{act}}(s')\,\mathrm{d}s' + c_T(d_2 - \tau_2 y_T). \quad (32)$$

Similarly the third term on the right hand side of Equation (30) simplifies to:

$$\int_{T+1}^{T+1+\tau_2 y_{T+1}} \mathbf{p}_1^{\text{act}}(s')\,\mathrm{d}s' + c_T \tau_2 y_{T+1}. \quad (33)$$

Substituting the expressions in Equations (32) and (33) back into Equation (30), making the approximation that $y_{T+1} \approx y_T$, simplifying, and then finally substituting the expression for $c_{\lfloor t \rfloor}$ from Equation (28) yields the right hand side of Equation (29).

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 (27) provides an approximate relationship between the active part of two solutions of the measurement constraints for two different settings of τ. Due to regularization and discretization, the final solution obtained by our algorithm will not exactly match Equation (27). It should hold approximately, however.

What does Equation (27) mean in terms of the final correction applied? First, note that only the active part of the solution is used in the correction process. Figure 4 illustrates how only the motion between T and $T + d = T + \tau(M-1)$ is used in the correction process. Second, note that if $\mathbf{c}_{\lfloor t \rfloor} = \mathbf{0}$ then Equation (27) would mean that exactly the same correction is applied.

The proof of this fact follows a similar argument to the one above. Suppose that $(\mathbf{x}, t_2)$, where $\mathbf{x} = (x, y)$, is a solution of Equations (9) and (10) for τ_2. Then we claim that $(\mathbf{x}, t_1) = (\mathbf{x}, r(t_2 - T) + T)$ is a solution of Equations (9) and (10) for τ_1. Because the spatial location $\mathbf{x}$ in both of these solutions is the same, our algorithm would apply the same correction. Note that, in both cases we set β using Equation (2); i.e. $\beta_1 = \tau_1(M-1)/2$ and $\beta_2 = \tau_2(M-1)/2$.

First consider Equation (9). Starting with:

$$t_1 \;=\; r(t_2 - T) + T \quad (34)$$

and substituting $t_2 = T + \tau_2 y$ because t_2 is a solution of Equation (9) yields:

$$t_1 \;=\; T + r\tau_2 y. \quad (35)$$

Using the fact that $r = \tau_1/\tau_2$ shows that $(x, y, r(t_2 - T) + T)$ is a solution of Equation (9) for τ_1.

Now consider the right hand side of Equation (10):

$$\mathbf{X} + \int_{T+\beta_1}^{t_1} \mathbf{p}_1^{\text{act}}(s)\,\mathrm{d}s. \quad (36)$$

Substituting the appropriate expressions for β_1 and t_1 yields:

$$\mathbf{X} + \int_{T+r\beta_2}^{T+r(t_2-T)} \mathbf{p}_1^{\text{act}}(s)\,\mathrm{d}s. \quad (37)$$

After substituting $s = r(s' - T) + T$ and $r\mathrm{d}s' = \mathrm{d}s$, this expression simplifies to:

$$\mathbf{X} + \int_{T+\beta_2}^{t_2} r\mathbf{p}_1^{\text{act}}(r(s'-T)+T)\,\mathrm{d}s'. \quad (38)$$

Substituting the expression for $\mathbf{p}_2^{\text{act}}$ from Equation (27) with $\mathbf{c}_{\lfloor t \rfloor} = \mathbf{0}$, and using the fact that $(\mathbf{x}, t_2)$ is a solution of Equation (10) for τ_2 and β_2 shows that Equation (38) simplifies to $\mathbf{x}$, which in turn shows that $(\mathbf{x}, t_1)$ is a solution of Equation (10) for τ_1 and β_1.

The difference in the two corrections is therefore entirely 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. See Equation (27).

In summary, theory shows that, with an incorrect value of τ (close enough that the effects of discretization, regularization, and the $y_{T+1} \approx y_T$ approximation are not too pronounced), the applied correction will only differ from the one that would have been obtained with the correct value of τ by a slightly different affine warp for each

frame. Although estimating τ wrongly may add a little global jitter to the output, our algorithm can be expected to be relatively robust in the sense that there is little danger of gross artifacts being added.

2.4.2 An Auto-Calibration Algorithm

We now describe an algorithm to calibrate τ from a short segment of video. No other information is required. Note, however, that if the chosen segment only contains constant (or no) motion, calibration is ambiguous. If the motion is constant, the rolling shutter simply introduces a skew in the video. It is impossible to detect this skew from the motion alone. The motion is still constant in the skewed video; the x-component is the same, the y-component will be slightly reduced or exaggerated depending whether the camera is moving up or down. Conceptually, the simplest way to calibrate a rolling shutter camera in the lab is to first capture an image of a scene with the camera static, and then capture a short video of the same scene with the camera undergoing a constant motion [15]. The rolling shutter parameter τ can then be estimated from an estimate of the skew between the first image and a frame in the video, and the motion in the video. The key element in this set-up is that the data contains imagery with two different motions (zero and constant.) For our auto-calibration process not to be ambiguous, we require the video to contain temporally varying motion. Generally speaking, the more jitter the better.

The analysis in Section 2.4.1 shows that if the value of τ is wrong, we can expect the corrected video to contain a small residual affine jitter from frame to frame. We attempt to detect and minimize this residual affine jitter, as follows. Note that although the analysis in Section 2.4.1 assumes a translation jitter model and makes several approximations, our auto-calibration algorithm is a reasonable approach in a much wider setting. Also note that, although our calibration 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.

We perform rolling shutter correction for a sampling of different values of $\tau \in [0, \frac{1}{M-1}]$ and compute optical flow across each output video. Denote the result:

$$\mathbf{F}_T(X,Y) \quad \text{for } T = 0, 1, 2, \ldots \tag{39}$$

We then compute a measure of how "translational" the optical flow is on average across the sequence. We first break the frames in the video into a number of patches P^i, where each patch denotes a subset of pixels. In our implementation, we used overlapping 15×15 pixel patches, spaced every 5 pixels in x and y, with a border of 30 pixels around each image. We then compute the median flow for each patch:

$$\mathbf{M}_T^i \;=\; \text{Median}_{(X,Y)\in P^i}(\mathbf{F}_T(X,Y)). \tag{40}$$

Next, we measure the median deviation of each flow from the median to give the following measure of how translational the motion is for that patch:

$$\text{Trans}_T^i \;=\; \text{Median}_{(X,Y)\in P^i} \left|\mathbf{F}_T(X,Y) - \mathbf{M}_T^i\right| \tag{41}$$

Finally, we compute the median value of this measure across the patches, and then the mean across the frames to yield the final measure that we optimize:

$$\text{Mean}_T(\text{Median}_i(\text{Trans}_T^i)). \tag{42}$$

We used a median across patches because some patches will lie across motion discontinuities or contain substantial parallax. We used a mean across frames because typically each frame has a number of outliers, but otherwise is reasonable. Note, however, that the details of our algorithm (the patch sizes, the sampling interval, the choice of medians or means) are somewhat arbitrary, and do not affect the results significantly.

Finally, we plot the measure in Equation (42) across τ, smooth the result slightly, and then perform the calibration by choosing τ to take the minimum value.

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 dramatic 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 (four frames before and four frames after the frame we are currently correcting.)

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 affine and

independent motion cases, 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. In this case, we use a single correspondence per row, obtained by median filtering the flow along each row (after subsampling every 10th column as we did in the affine and independent motion cases.) We experimented with $K > 1$ but only found a small benefit. The results in this paper all use $K = 1$.

We experimented with different sampling rates for the motion parameters. 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. In the affine case, we sample $\mathbf{q}$ 15 times per frame, adding an additional 60 unknowns per frame. We subsample $\mathbf{m}_i^{\text{lf}}(\mathbf{x})$ spatially every 10 rows and every 10 columns. We use bilinear interpolation to upsample $\mathbf{m}_i^{\text{lf}}(\mathbf{x})$ in the algorithm; i.e. in Equations (21) and (25).

With the translational model we use the regularization weight $\lambda = 300$. In the affine and independent motion cases, the system has more flexibility and so we use $\lambda = 600$. The affine regularization weight is $\delta = 400,000$ (the units are different from λ.) The independent motion regularization weights are $\gamma = 10$ and $\epsilon = 1$. Most of our test sequences were downloaded from the web and so we do not know the camera details. For all sequence, we estimated τ using the algorithm in Section 2.4.2. 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. This synthetic video also allows us to compare our motion estimates with ground-truth. In Figure 6(a) we include one frame from the video in `city.mp4`. In the top left we show the input rolling shutter video which was generated synthetically using the translation model in Equation (3). The ground-truth high-frequency motion that we applied was generated with a dynamical model that periodically applies random accelerations to the current motion, but with a slight bias towards moving the camera back towards it position at the top of the first frame. Visually, the resulting artifacts are similar to those in the real data in Section 3.2.

In the bottom left of Figure 6(a) 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 because each frame has been re-rendered as though all the pixels were captured at the same time.

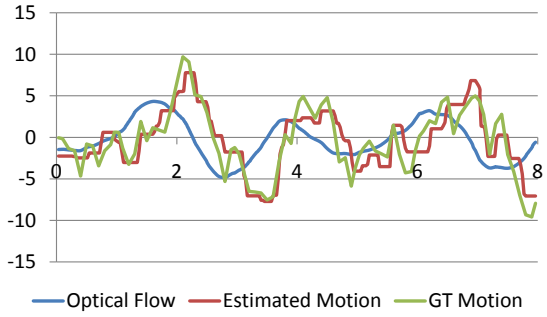
In Figures 6(b) and (c) we compare the input optical flow (median filtered across each row), the estimated high-frequency motion, and the ground-truth high-frequency motion. We plot the value of the flow and motion on the y-axis. On the x-axis we plot time, which in the case of the optical flow corresponds 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 of approximately half a frame between the optical flow and the real camera motion.

We also include a video in `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 skew correctly; i.e. when we apply temporal super-resolution to constant motion, no unexpected high-frequency motion is generated and the result is still approximately constant motion. For super-resolution in the image domain [3], this test corresponds to checking that an algorithm applied to a constant image still results in a constant image. Note, however, that for both results in this section, the rolling shutter algorithm is applied to the result of the real optical flow algorithm, not to synthetically generated correspondences.

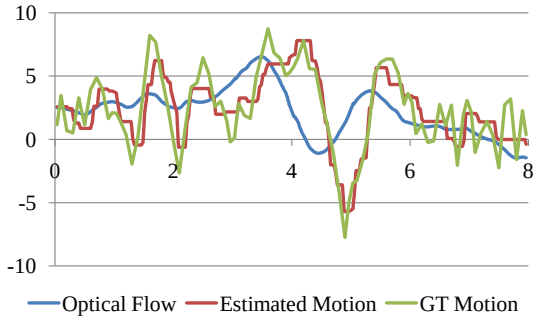
3.2 Translational Model

Our main evaluation consists of a set of qualitative results on real videos. In `skool.mp4`, `vegas1.mp4`, `vegas2.mp4`, `bike.mp4`, `reverse.mp4` and `race.mp4` we present results on 6 videos. In Fig-

(a) One Frame From a Synthetically Generated Movie

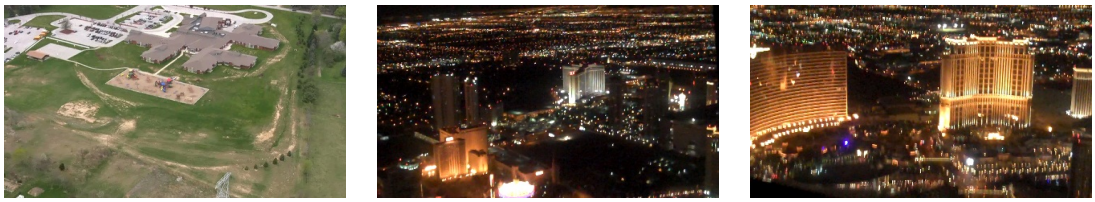


(b) Motion in x Direction



(c) Motion in y Direction

Figure 6: **An Illustrative Example**: We generated a synthetic sequence to illustrate our algorithm, and allow a comparison with ground-truth. (a) One frame from the video in `city.mp4`. Our algorithm first computes the optical flow (a, bottom left). We then compute a corrected video without the non-rigid wobble artifacts, but which still contains *global* jitter (a, bottom right). We then stabilize the result (a, top right.) (b) and (c) A comparison of the low-frequency optical flow with the estimated and ground-truth high-frequency motion. Note how the input optical flow field is far lower frequency than the ground-truth motion. There is also a phase shift of roughly half a frame. The estimated motion approximates the ground-truth motion far better than the input optical flow.



(a) skool.mp4 (b) vegas1.mp4 (c) vegas2.mp4

(d) bike.mp4 (e) reverse.mp4 (f) race.mp4

Figure 7: **Inputs to Qualitative Evaluation**: Our main evaluation consists of a set of qualitative results on real videos. In skool.mp4, vegas1.mp4, vegas2.mp4, bike.mp4, reverse.mp4 and race.mp4 we present results on 6 videos. In this figure, we include the first frame of each input video.

ure 7, we include the first frame of each input video. skool.mp4 contains results on an aerial sequence, shot from a helicopter. The movies vegas1.mp4 and vegas2.mp4 also include results on aerial videos. These two results show the robustness of our algorithm to low light conditions, saturation, and motion blur (which we do not attempt to remove.) The video bike.mp4 contains results on a video captured on a fast moving motorbike. It shows that our algorithm is robust to very noisy input, where the computed flow is very noisy. The video reverse.mp4 shows a more subtle case, where a motorbike is being pulled backwards. This video is more typical of rolling shutter wobble in hand-held videos. This video shows that our algorithm can handle such subtle cases, and also performs reasonably in the presence of some rotational motion. Finally, race.mp4 contains 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. This result shows that our algorithm does not lead to complete video corruption, even when optical flow completely fails.

All of the videos in skool.mp4, vegas1.mp4, vegas2.mp4, bike.mp4, reverse.mp4 and race.mp4 are formatted the same. The layout is illustrated in Figure 8, which includes one frame from skool.mp4. In the top left frame 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 also implemented the algorithm in [7] and the morphing algorithm in [5]. Empirically, we found the algorithm in [7] to perform better than the one in [5]. We only present the results for the algorithm in [7] in the bottom right of Figure 8 and the corresponding videos in the supplementary material. 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

Figure 8: **Qualitative Comparison**: One frame from the video in `skool.mp4` to illustrate the layout in all our main qualitative comparisons in `skool.mp4`, `vegas1.mp4`, `vegas2.mp4`, `bike.mp4`, `reverse.mp4` and `race.mp4`. In the top left, we include the original rolling shutter video. 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).

the motion. While [7] corrects some artifacts, it does not remove all of the wobble.

3.3 Affine Model

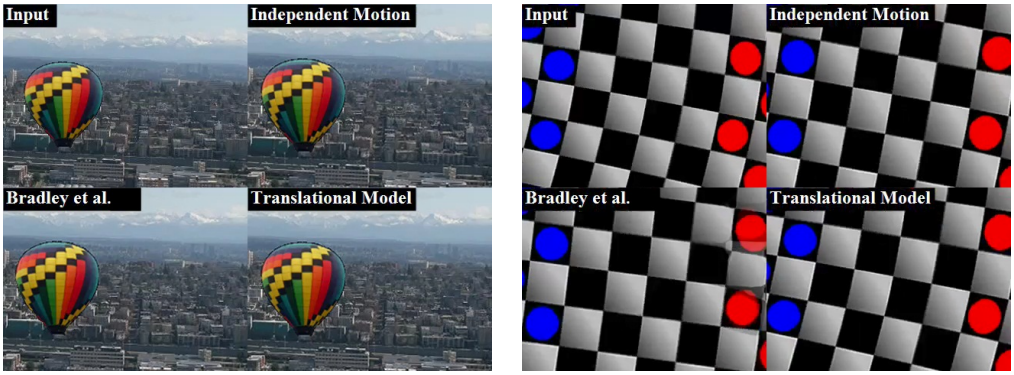
In `office.mp4` we include an example where the translational jitter model of Section 2.1 is insufficient. The video compares the results obtained with the translational model (bottom right) and the results obtained with the affine jitter model described in Section 2.2 (top right). Figure 1(a) contains one input frame from this video and Figure 1(b) contains the corresponding output frame obtained with the affine model. As can be seen in the video in `office.mp4`, the obliquely sloping office building is far better modeled by the affine jitter model than the translational model. In our experience, however, the results in `office.mp4` are somewhat unusual. It is relatively rare for the affine model to result in a significantly better correction than the translational model.

3.4 Independent Motion

In `balloon.mp4`, `toggle.mp4` and `checker.mp4` we compare the independent motion algorithm of Section 2.3 with the translational model of Section 2.1 and the morphing algorithm of Bradley *et al.* [5]. Figure 9(a) contains one frame of `balloon.mp4`, which contains results on a synthetic video of a balloon moving across the sky, imaged by a translationally jittering camera. In the video we include the input (top left), the output of our independent motion algorithm (top right), the output of our translational algorithm (bottom right), and the results obtained by Bradley *et al.* [5] (bottom left.)

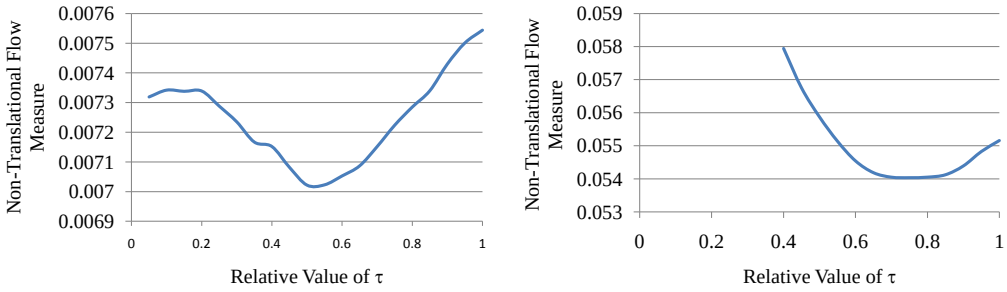
As can be seen in `balloon.mp4`, the morphing algorithm of Bradley *et al.* [5], which is specifically designed to handle independent motion, cannot handle high-frequency jitter. Also note that the translational model is robust to the independent motion, through the median filtering of the flows and the L1 norms. It corrects the video very well. Finally, note that the residual skew on the balloon is largely imperceptible in `balloon.mp4`. The balloon is moving too quickly and there is no frame of reference against which to perceive the skew. In `toggle.mp4`, we toggle between the ground-truth frames and the ones estimated by the translational and independent motion algorithms. In this video, the residual skew of the balloon without the independent motion algorithm is readily apparent.

In `checker.mp4` and Figure 9(b) we include an example of a rotating checkerboard, similar to one of the examples presented in [5]. Our example also contains high-frequency jitter not present in the example in [5]. Rotational motion is a useful tool when evaluating independent motion modeling because errors make straight lines appear curved. Errors are then more perceptible. As can be seen in `checker.mp4` and Figure 9(b), the translational algorithm of Section 2.1 is unable to correct the sequence very well. Significant curvature remains in the output. The algorithm in [5] performs fairly well, but the unmodelled jitter leads to gross artifacts and some residual curvature in some of the frames. On the other hand, our independent motion model is able to correct the entire video very well.



(a) `balloon.mp4` (b) `checker.mp4`

Figure 9: **Independent Motion Results**: (a) One frame from the video in `balloon.mp4`. In this video, the balloon moves across the sky in front of a jittering camera. Without the independent motion model, the balloon has a residual skew which is not corrected. The video in `toggle.mp4` toggles between the estimated frames and the ground-truth, clearly showing this residual skew. In `balloon.mp4` we also compare with the algorithm in Bradley *et al.* [5], which fails to remove the camera jitter fully. (b) One frame from the video in `checker.mp4`. This video contains a rotating checkerboard imaged by a jittering camera. The results show that the translational model is insufficient to correct the rotational motion, that the algorithm of Bradley *et al.* [5] leads to various artifacts and fails to compensate for the jitter fully, but that our independent motion model corrects the video far better.



(a) Calibration on Online Resource 1 (b) Calibration on Online Resource 3

Figure 10: **Quantitative Calibration Results:** (a) Auto-calibration results on a version of `city.mp4` where the ground-truth relative value of τ is 0.5. The estimated minimum is very close to 0.5 as it should be. (b) Auto-calibration results on `skool.mp4`. We do not know the real value for this sequence, as the video was downloaded from the web. In `skool_cal.mp4` we present a comparison of the correction results obtained with the relative value of $\tau = 0.25$, 0.5, 0.75, and 1.0. This video confirms that the calibration result of 0.75 is reasonable (the results for $\tau = 0.75$ are slightly better than the others) and illustrates the robustness of our algorithm (the results for the other settings of τ are visually not much worse.)

3.5 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 10(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 10(b) we present calibration results for the `skool.mp4` video. We downloaded this video from the web and so do not know the real value of τ. In `skool_cal.mp4` we present a qualitative comparison of the affect of varying the relative value of τ (i.e. multiplied by $M-1$). These results confirm two things. First, the calibrated relative value of $\tau = 0.75$ does appear to be reasonable. If anything, the results for $\tau = 0.75$ are slightly better than the results for the other settings. The difference in performance is quite small, demonstrating that our algorithm is relatively insensitive to the exact choice of τ. These results validate the analysis in Section 2.4.1.

3.6 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, 79.3 seconds per frame for the affine 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 obtained by computing the correction on a half-size video, and then applying to the full resolution video. There is little difference, indicating that speed-ups are possible. To obtain real-time performance, however, a far faster flow algorithm would be needed. It may also be necessary to replace the L1/Linear Programming algorithm with something more efficient. The investigation of such trade-offs is deferred to future work.

4 Conclusion

We have presented an algorithm to remove rolling shutter wobble in video. Our algorithm uses a form of temporal super-resolution to infer the high-frequency motion of the camera from optical flow. We extended our algorithm to use an affine motion model and to model low-frequency independent motion. Empirically, the improvements obtained using these extensions are most perceptible when comparing with the ground-truth on synthetic sequences. Given the increased computational burden and sensitivity to errors in the input optical flow, these extensions are probably more of theoretical interest at this time. We showed both analytically and empirically that our algorithm is robust to the setting of the image formation parameter τ. We also presented an auto-calibration algorithm that can estimate this parameter from a short segment of the video.

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, regularization, and optimization algorithm, both to improve quality and speed. Another possibility is to explore the direct estimation of the motion model parameters; i.e. without first estimating optical flow or feature correspondences.

Acknowledgements

The first author 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]. A preliminary version of this paper appeared in the IEEE Conference on Com-

puter Vision and Pattern Recognition [2]. Finally, thanks to Russ Andersson of Andersson Technologies LLC (http://www.ssontech.com/) for providing the "skool" sequence used in `skool.mp4` and `skool_cal.mp4`.

References

[1] O. Ait-Aider, A. Bartoli, and N. Andreff. Kinematics from lines in a single rolling shutter image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[2] S. Baker, E. Bennett, S. Kang, and R. Szeliski. Removing rolling shutter wobble. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

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

[4] M. J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision, Image Understanding*, 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 the International Workshop on Projector-Camera Systems*, 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 Transations on Image Processing*, 17(8):1323–1330, 2008.

[8] F. Liu, M. Gleicher, H. Jin, and A. Agarwala. Content-preserving warps for 3D video stabilization. *ACM Transactions on Graphics*, 28(3), 2009.

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

[10] M. Meingast, C. Geyer, and S. Sastry. Geometric models of rolling-shutter cameras. In *Proceedings of the International Workshop on Omnidirection Vision, Camera Networks, and Non-Classical Cameras*, 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 Transactions on Pattern Analysis and Machine Intelligence*, 27(4):531–545, 2005.

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

[16] Wikipedia, The Free Encyclopedia. Rolling shutter. http://en.wikipedia.org/wiki/Rolling_shutter.