In-Camera All-Digital Video Stabilization

Aaron Deever; Eastman Kodak Company; Rochester, NY

Abstract

Video sequences captured with hand-held digital still cameras often contain unwanted motion caused by hand jitter. In this paper, we investigate the problem of video stabilization, and in particular, focus on in-camera, all-digital video stabilization. Algorithms for global camera motion estimation and jitter calculation are proposed. Computational constraints imposed by an in-camera solution are also discussed.

Introduction

Many digital cameras are capable of capturing video sequences as well as still images. These video sequences often contain unwanted motion caused by hand jitter. This unwanted motion is distracting, and can also reduce visual quality and encoding efficiency.

Several different approaches to jitter removal exist, ranging from optical to electronic to digital. Optical solutions may use an oscillating gyroscope and rotating prism lens as part of a mechanism to detect and correct for angular velocity in the camera [1]. Although effective at stabilization and at reducing motion blur, optical stabilization increases camera cost.

Stabilization may also be achieved electronically through the use of motion-sensing transducers, which detect actual camera motion. The jitter component of this motion can be computed and compensated for by selecting an appropriately offset region from an oversized CCD or CMOS imaging plane [1].

Stabilization can also be accomplished entirely in the digital domain. With digital stabilization, true camera motion must be estimated from the captured video stream. This approach has low cost because it is entirely algorithmic and is implementable as a firmware solution. It encounters increased performance challenges relative to optical or electronic stabilization, however, because the computational resources available for stabilization are usually limited, and camera motion estimation from arbitrary video sequences can often be confused by content.

The workflow for digital stabilization is shown in Figure 1. The first step is to estimate the motion between frames. This is followed by trajectory estimation, which computes an estimate of the *desired* camera motion (e.g., a panning motion). Jitter is estimated based on the overall motion and desired camera motion estimates, and is then compensated for through an image shift or warp function.

Figure 1. *The workflow of a typical digital video stabilization algorithm.*

Digital stabilization can be performed using software off-line after a video sequence has been captured. This approach has the advantage that significant computational resources can be applied to the problem, allowing sophisticated motion estimation and jitter correction algorithms. Off-line stabilization has several disadvantages, however. It may introduce additional artifacts caused by multiple compression/decompression cycles, and also results in the loss of some image resolution. Off-line stabilization also requires an extra step of user interaction, which increases overall complexity. In this paper, we focus on in-camera, firmware-based stabilization, in which the captured video data is stabilized prior to compression and storage.

Of initial importance in a stabilization solution is deciding how the final crop and/or warp will be performed, because this decision affects the motion estimation models that may be considered. Usually the image information is stored in a buffer that is arranged in raster scan fashion. The easiest way to move this data around is to perform an integer shift of the data horizontally and vertically. This shift introduces no distortions in the image data and can be done very quickly. A more complicated adjustment might be a noninteger shift horizontally and vertically, requiring an interpolation step. This can result in blurred edges and is more expensive to compute. There are increasingly complicated transformations such as affine or perspective transforms that would also require a warp of the entire frame. Non-real-time software systems have the luxury of using these motion models, but digital still cameras often spend most of their compute cycles in video mode with compression tasks; therefore, limited resources are available for stabilization.

Another major point of practical consideration is whether the stabilization algorithm is performed on Bayer data directly from the image sensor, or on YCbCr data after some image processing has taken place. Bayer data is typically of higher resolution than the subsequent YCbCr data, and thus offers the opportunity for finer precision stabilization. This issue does not directly affect the stabilization algorithms, however. Typically, motion estimation algorithms operate on luminance data. With Bayer data, these algorithms could easily be applied to the green image data instead. The computational cost of storing and manipulating high-resolution CFA data can be prohibitive, however.

In order to allow for jitter correction, it is desirable to have an image sensor providing data with a larger area than will be saved in the video sequence. This concept is illustrated in Figure 2, in which the video resolution is less than the resolution captured by the image sensor, such that a buffer zone exists both horizontally and vertically. This allows the stabilization algorithm to extract an appropriately shifted portion of the image sensor data, based on the computed translational jitter.

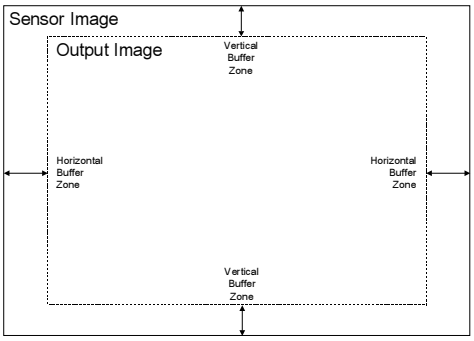

Figure 2. *The buffer zone concept for an image with a larger sensor area than will be saved in the video sequence.*

Note that there is a limit to the amount of jitter that can be compensated before reaching the edge of the image sensor data.

While it is preferred to have an image sensor with greater resolution than the final video sequence, it is possible to perform digital stabilization in the case that the image sensor does not provide any buffer zones. In this scenario, a shrunken region of the image sensor is treated as the desired video resolution, and stabilization is performed as illustrated in Figure 2. Subsequently, the shrunken video sequence can be interpolated back to the original image sensor resolution. The interpolation step increases complexity, however, and also may degrade image quality.

In the remainder of this paper, we assume that the final image shift will be restricted to an integer translational offset, and focus on the remaining aspects of the stabilization algorithm: motion estimation and jitter calculation. In the next section, we briefly review motion stabilization techniques. This is followed by a description of an algorithm for jitter calculation. Experiments and results are subsequently detailed, and a summary completes the paper.

Motion Estimation

The majority of previous work in digital stabilization utilizes some form of block-matching for motion estimation. Block-matching involves dividing an image into a collection of blocks, and for each block finding the best matching block in the previous image. A general overview of block-matching for motion estimation can be found in [2]. When exhaustive searches are used to find the best match for each block, this technique is prohibitively complex. Several improvements can be incorporated to improve the efficiency of this algorithm, however. Hierarchical searches perform a coarse-to-fine estimate of the motion, and only require that a fraction of the potential solutions be considered. Gray-coded bit planes and edge maps have been proposed as methods by which to convert 8-bit image data into a single bit for each pixel [3,4]. These approaches allow blocks to be compared through bit operations rather than through more expensive subtraction operations.

Once a motion estimate has been obtained for each block, a set of rules must be applied to convert these local estimates into a single global estimate of the motion. Because block-based motion estimation obtains local motion estimates from different regions throughout the image, it can be very robust to independent moving objects within a scene. Local estimates may be eliminated if they are considered unreliable due to causes such as the block containing repeating patterns or very few edges [5]. Once the local estimates have been pruned such that only reliable estimates remain, typically the median or mean is chosen as the global motion estimate [6].

As an alternative to block-based motion estimation, the technique of integral projections can be used to obtain a fast, robust estimate of the dominant global translational motion between two frames [7,8]. Integral projections operate by projecting a two-dimensional image onto two one-dimensional vectors: one horizontal and one vertical. This is achieved by summing the elements in each column to form a vertical projection (used to compute the horizontal motion estimate), and by summing the elements in each row to form a horizontal projection (used to compute the vertical motion estimate), as illustrated in Figure 3. This process is repeated for a second image as well.

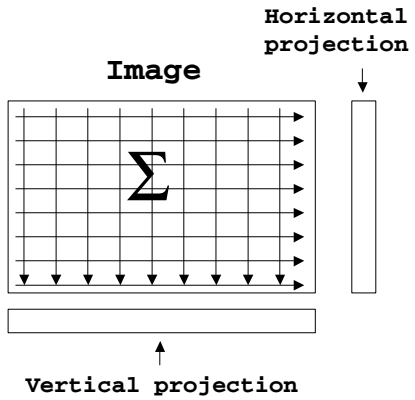

Figure 3. *Integral projections. The two-dimensional image is converted into two one-dimensional projection vectors by summing along rows, and down columns.*

The vertical projection vectors from the two images are correlated to find the offset providing the best match. Typically an L_1-norm (sum of absolute differences) is used as the error metric, and the offset with the lowest error is chosen as the horizontal motion between the two frames. The process is repeated independently with the horizontal projection vectors to determine a vertical motion estimate.

Computational speedups are possible using two varieties of subsampling. The first subsampling reduces the number of samples included when summing a given row or column. Usually it is desirable to have at least 100 samples included in each sum, if possible. Excessive subsampling can result in aliasing that decreases the accuracy of the motion estimate. The second subsampling involves reducing the precision of the motion estimation by only summing data for a subset of the rows or columns. Some precision can be reacquired by interpolating the derived projection vectors prior to correlating them at various offsets. Normally this subsampling is restricted to a factor of two, which is recovered by interpolation of the projection vectors.

Jitter Calculation

Once a motion estimate has been computed, it remains to determine what component of that motion is desired, because of a camera pan, for example, and what component of the motion is caused by camera jitter. In the simple case when the desired motion is known to be zero, all of the estimated motion can be treated as jitter and removed from the sequence. In general, however, there may be some desired camera motion along with the undesirable camera jitter. Typically it is assumed that any desired camera motion is of very low frequency, no more than 1 or 2 Hz. Many studies have shown hand shake to commonly occur between 2–10 Hz [9,10]. Low-pass temporal filtering can thus be applied to the motion estimates to eliminate the high-frequency jitter information while retaining any intentional low-frequency camera motion.

In addition to having a specific frequency response that eliminates high-frequency jitter information, the ideal low-pass filter for video stabilization also needs to have minimal phase delay. Ideally, a symmetric, zero-phase linear filter is used for stabilization. For a given frame being stabilized, however, this requires motion estimates from both previous frames and future frames. This is possible for stabilization applications that are run off-line. For in-camera stabilization, however, some phase delay is unavoidable. This is a result of the constraint that the stabilized video should be displayed on the back of the camera with minimal time lag. This necessitates causal filtering, which results in non-zero phase delay.

Most previous work in filter design for stabilization has derived from the work of Uomori [10]. In his work, Uomori combined the steps of filtering and jitter accumulation into one formula, given by the equation:

$$A[n] = \alpha A[n-1] + v[n], \tag{1}$$

where *A[n]* is the accumulated jitter for frame *n*, *v[n]* is the computed motion estimate for frame *n*, and α is a dampening factor with a value between 0 and 1 that is used to steer the accumulated jitter toward 0 when there is no motion. Note that the accumulated jitter is tracked separately for the *x* direction and *y* direction, and the term *v[n]* generically represents motion in either direction.

Although Uomori describes the different frequency responses that can be achieved through this system by varying the value of α, neither his work nor any of the subsequent papers adequately describe the phase response of the system. It turns out that the phase delay is suboptimal, and can be improved by analyzing and modifying the formula given by Equation (1).

Let *A* and *v* be defined as above, and let *s* be the smoothed motion estimate, defined as the original motion estimate minus the computed jitter. Then

$$s[n] = v[n] - (A[n] - A[n-1]), \tag{2}$$

where *A[n]-A[n-1]* represents the individual jitter for frame *n* relative to frame *n-1*. Thus:

$$\begin{aligned} s[n] &= v[n] - (\alpha A[n-1] + v[n] - A[n-1]) \\ &= (1-\alpha)A[n-1] \end{aligned}. \tag{3}$$

Then a recursion formula for *s* is given by the following:

$$\begin{aligned} s[n] &= (1-\alpha)A[n-1] \\ &= (1-\alpha)(\alpha A[n-2] + v[n-1]) \\ &= \alpha s[n-1] + (1-\alpha)v[n-1] \end{aligned}. \tag{4}$$

In this equation, the smoothed motion estimate for frame *n* is independent of the original motion estimate for frame *n*. It relies only on the previous smoothed motion estimate as well as the original motion estimate for frame *n – 1*. This introduces an unnecessary phase delay in the filter. A filter with identical frequency response but improved phase response can be constructed by simply using the current motion estimate when forming the smoothed estimate:

$$s[n] = \alpha s[n-1] + (1-\alpha)v[n]. \tag{5}$$

It can be shown that this filter results in the accumulation formula:

$$A[n] = \alpha A[n-1] + \alpha v[n]. \tag{6}$$

A comparison of the phase responses for the original Uomori and improved filters is illustrated in Figure 4.

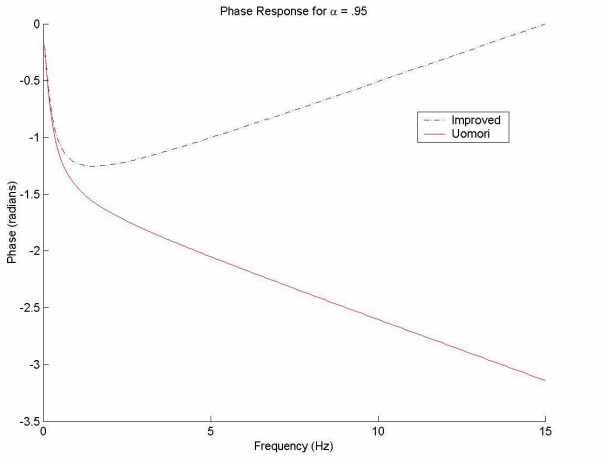

Figure 4. *Phase response for Uomori and improved filters. 30 frames per second is assumed, so that the maximum frequency is 15 Hz.*

As will be shown in the following section, the improved accumulation formula given by Equation (6) does a better job of retaining intentional panning motion, and results in decreased accumulated jitter. Visually, the improved accumulation formula corresponds to less of a trailing effect in the jitter-corrected video when an intentional pan occurs.

The accumulated jitter is clipped according to the maximum available buffer area of the image sensor. This clipping is necessary because of the limited size of the sensor, but is desirable even without this constraint to prevent the jitter-corrected video from falling too far behind during an intentional pan, as a result of phase-delay misclassification of motion as jitter.

Experiments and Results

Simulations were performed on a QVGA-resolution video captured at an indoor hockey rink. The video contained significant panning motion as well as jitter. Figure 5 shows the effect of the

improved phase response filter on the accumulated horizontal jitter for the sequence.

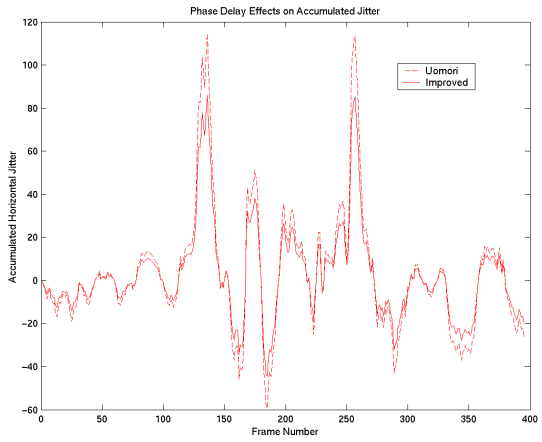

Figure 5. *Accumulated horizontal jitter for the hockey video sequence, based on the accumulation formula given in Uomori, and compared to the improved accumulation formula with reduced phase delay. The improved accumulation formula did a better job of identifying intentional panning motion, and resulted in reduced accumulated jitter.*

The improved filter performed particularly better in regions of panning motion, during which some motion was misclassified as jitter.

Figure 6 illustrates the general ability of the stabilization algorithm to remove jitter from the sequence. In this figure, only the first 100 frames of the hockey video sequence were considered, corresponding to a period of little intentional motion.

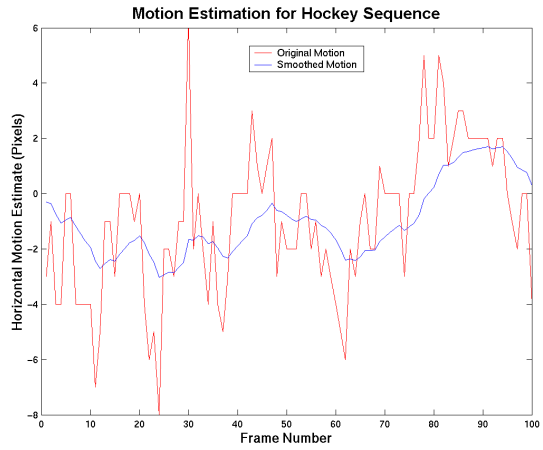

Figure 6. *Horizontal motion estimation for the first 100 frames of the hockey sequence. The original motion estimate was restricted to pixel accuracy, based on subsampling of the projection vector precision by a factor of 2, followed by interpolation to reacquire pixel precision. The smoothed estimate of the motion was obtained using the improved version of Uomori's recursive filter.*

The numerical jitter removal illustrated in Figure 6 corresponded to a stabilized video with significantly improved visual quality.

Although for brevity additional results are not described in detail, the proposed stabilization algorithm was successful at reducing jitter and improving visual quality for a variety of video sequences.

Summary

In this paper we discuss motion estimation and filtering algorithms for digital video stabilization, and in particular, derive a causal filter with improved phase response relative to existing approaches. We also discuss trade-offs associated with performing stabilization in-camera versus off-line stabilization of a previously encoded video sequence.

References

[1] B. Schweber, "How It Works: Image Stabilization Shows Diversity of Engineering Approaches," www.ednmag.com, October 26, 2000.

[2] C. Stiller and J. Konrad, "Estimating Motion in Image Sequences," *Signal Processing Magazine*, pp. 70 (1999).

[3] S.J. Ko et al., IEEE Transactions on Consumer Electronics, 45(3), pg. 598 (1999).

[4] J. Paik, Y. Park and D. Kim, IEEE Transactions on Consumer Electronics, 38(3), pg. 607 (1992).

[5] J. Paik, Y. Park, and S. Park, IEEE Transactions on Consumer Electronics, 37(3), pg. 521 (1991).

[6] K.S. Choi et al., International Conference on Consumer Electronics, pg. 246 (2000).

[7] K. Ratakonda, IEEE International Symposium on Circuits and Systems, 4, pg. 69 (1998).

[8] K. Sauer and B. Schwartz, IEEE Transactions on Circuits and Systems for Video Technology, 6(5), pg. 513 (1996).

[9] R. Stiles, Journal of Applied Physiology, 40(1), pg. 44 (1976).

[10] K. Uomori et al., IEEE Transactions on Consumer Electronics, 36(3), pg. 510 (1990).

Author Biography

Aaron Deever received his B.S. degree in mathematics and computer science from The Pennsylvania State University, and a Ph.D. degree in applied mathematics from Cornell University. He is currently a Research Scientist at Eastman Kodak Company, Rochester, NY, where he works on image and video processing applications.