A Virtual Tripod for Hand-held Video Stacking on Smartphones

Erik Ringaby and Per-Erik Forssén
Department of Electrical Engineering, Computer Vision Laboratory
Linköping University, Sweden
{ringaby,perfo}@isy.liu.se

Abstract

We propose an algorithm that can capture sharp, low-noise images in low-light conditions on a hand-held smartphone. We make use of the recent ability to acquire bursts of high resolution images on high-end models such as the iPhone5s. Frames are aligned, or stacked, using rolling shutter correction, based on motion estimated from the built-in gyro sensors and image feature tracking. After stacking, the images may be combined, using e.g. averaging to produce a sharp, low-noise photo. We have tested the algorithm on a variety of different scenes, using several different smartphones. We compare our method to denoising, direct stacking, as well as a global-shutter based stacking, with favourable results.

1. Introduction

In this paper we propose an algorithm that can capture sharp, low-noise images in low-light conditions on a hand-held smartphone. Recent smartphone models such as the Apple iPhone5s, Acer Liquid S2 and Samsung Galaxy Note 3 have the ability to acquire bursts of high resolution images at a high rate. For smartphones equipped with a gyroscope sensor, such image bursts may be aligned or *stacked* at low cost, using sensor data. The stacked frames can then be fused into a single (less noisy) frame using e.g. averaging, median or bilateral filtering. This enables hand-held acquisition of sharp, low-noise images with long exposure times, without the requirement of additional mechanical image stabilisation hardware.

Video stacking on a smartphone currently requires a tripod, due to the *rolling shutter* (RS) distortions that appear during hand-held capture. In this paper we introduce an RS based correction, that allows also images from hand-held video to be stacked.

An illustration of the proposed approach is given in Figure 1. Instead of using one long exposure, with resultant blurring, many short exposures are used in sequence. When the photographer has a *static aim* (i.e. tries to aim at a fixed point in space), these individual exposures tend to have blur smears in a random distribution of directions. This means that when the frames are aligned (using gyroscope readings and image feature tracking), we obtain an effective *point spread function* (PSF) that is much more compact than one from a single long exposure.

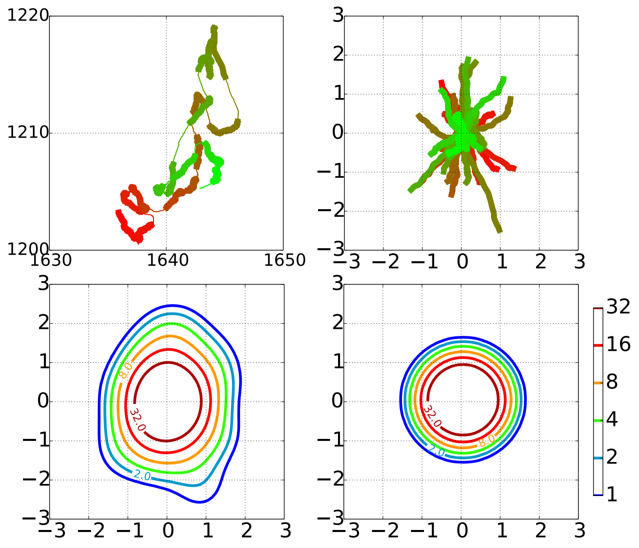

Figure 1. Illustration of video stacking idea. Top left: Trace of central pixel of an iPhone5 during hand-held capture of 20 frames at 14 fps, with 40 msec exposure time. The motion has been recorded with an L3G4200D gyroscope at 800Hz, and colours indicate time, ranging from red to green. Thick segments indicate individual exposures. Top right: Alignment of the exposure segments. Bottom left: Iso contours of the effective PSF (scaled to sum to 255), obtained by convolving the aligned exposure segments with the stationary PSF (here assumed to be a Gaussian with $\sigma = 0.5$). Bottom right: corresponding iso contours for a Gaussian with $\sigma = 0.5$.

1.1. Related work

Besides the stacking approach used in this paper, there are several other approaches to low-light image capture.

One is to use pairs of flash and no-flash images, see e.g. [14] and [3], and another is to use a pair with one blurry and one noisy image, e.g. [22]. As these approaches rely on accurate alignment of frames, they could also benefit from the rolling shutter aware alignment procedure proposed here, if they were to be used on mobile devices.

Another related approach is video denoising. Here the exposure time is set low enough to obtain sharp, but noisy images, and then spatiotemporal denoising [11, 15] is performed. These methods find correspondences across several frames, using e.g. dense optical flow, and approximate KNN search, and this makes them infeasible to implement on a smartphone.

A much faster, but less accurate approach than video denoising is to denoise a single frame captured with a short exposure, using e.g. anisotropic diffusion [20]. We will compare to single frame denoising in the experiment section.

Single frame deblurring using inertial sensors (INS) has been investigated in [8]. Recently this has also been extended to use a rolling shutter camera model [19]. Such algorithms are iterative in their nature, as they obtain the final sharp images using e.g. Richardson-Lucy style deconvolution [16, 13]. If the sensor biases also need to be found, this requires a second optimisation loop outside the first one [8]. In general, these algorithms are either relatively efficient, but prone to ringing artifacts, or very expensive if complex priors are employed.

Single frame deblurring without INS is also an option, see e.g. [21], and the recent approach in [23]. In addition to performing deconvolution, these approaches also need to find the point-spread functions in each image location, and as this is typically done using a second optimisation loop outside the first one, these methods are an order of magnitude more expensive than methods that use INS data.

As nearly all digital video recording devices use rolling shutters, the video stacking is related to video stabilisation under rolling-shutter [17, 5]. Stabilisation of RS video has also been done using inertial sensors [6, 9]. Basically, the problem in video stacking is a video stabilisation, where the desired camera trajectory is a single point in space with static aim. In order not to introduce blurring however, stacking has a much higher requirement on stabilisation accuracy than video stabilisation. For high accuracy we use the cumulative quaternion B-splines introduced by Kim et. al [10], to interpolate the gyro samples. These splines were also recently used in an optimisation framework for SLAM in [12], by minimising the reprojection errors on tracked features and sensor data. Compared to [12] our algorithm is many orders of magnitude faster, as our optimisation only needs to solve for 4 unknowns, instead of several thousands.

Stacking has been popular for quite some time in astrophotography, and the idea actually originates here [1]. Here the camera is typically mounted on a telescope that tracks slow motion (e.g. of the moon, or the celestial sphere), and long exposure times are used. The motion to be compensated for is thus the residual of the tracking and the actual motion, and not the complex atmospherical aberrations observed at short time-scales.

Recently stacking using inertial sensors has been introduced by compact camera manufacturers, in e.g. Sony Cyber-Shot DSC RX100 [2]. These devices use the motion sensors to select a few frames (up to 6 for Sony) with low amounts of motion. These are then stacked with global frame alignment (as the camera appears to use the mechanical shutter this is justified), using inertial sensors. In contrast, the method in this paper uses a rolling shutter distortion model, and is able to make use of *all* frames in the acquisition interval. It thus has a much better light collection efficiency, and better noise suppression.

1.2. Video stacking

The method proposed in this paper makes use of the built-in gyro sensor on a smartphone, and a sparse set of tracked image features to stack frames acquired using rolling shutter style exposure. The stacked frames can then be fused into a single (less noisy) frame using e.g. averaging, median or bilateral filtering.

For low-noise low-light photography, we should collect as much light as possible during the time the shutter is open. This means that for stacking, the *light collection efficiency*, e, is an important performance metric. This measure is approximately equal to the product of the shutter speed s in seconds, and the frame rate r in Hertz. For an N frame stack it is defined as the effective exposure time $s_e = sN$ divided by the time required to acquire the stack $T = (N-1)/r + s \approx N/r$.

$$e = s_e/T \approx sr. \tag{1}$$

For a single frame we always get $e = 1$, but e.g. a shutter speed of $s = 1/30$ sec and $r = 20$ fps, gives us $e \approx 0.67$.

Just like in classical photography, we have a built-in trade-off here: in order to eliminate motion blur, the shutter speed s should be short, but in order to collect more light it should be as long as the frame rate permits.

In order to improve the light collection efficiency (1) we will in general allow some motion blur. The rationale for this is illustrated in Figure 1. When the photographer has a static aim, the blur directions in consecutive images will be randomly distributed (see Figure 1 and 4), and this means that the final effective blur kernel will be much smaller than the smear in individual frames. The example in Figure 1 shows the effective PSF for stacking of 20 frames with 40 msec exposure time. Even though the smear length is about 3 pixels in each frame, the effective PSF is similar to a Gaussian of $\sigma = 0.5$.

2. Motion Model

We make use of a motion model that consists of a time continuous 3D rotation, and a frame global 3D translation. These models are estimated and applied in corrective fashion, one after the other. Such an approach normally requires *alternating optimisation*. The reason this works in one shot here is that the rotation model makes use of gyro sensors to estimate the rotation, and visual tracking to estimate the gyro bias and the time delay between gyro and camera. As the gyro sensors only sense rotation and not translation, the translation will not interfere with the rotation compensation, and the two models need only to be estimated once.

2.1. Rotation model

We use the 3D rotation model introduced in [4]. In this model, a point in the first frame, expressed in homogeneous coordinates as $\mathbf{x} = (x_1\ x_2\ 1)^T$, is related to its position in a subsequent frame $\mathbf{y} = (y_1\ y_2\ 1)^T$ according to:

$$\mathbf{x} \sim \mathbf{K}\mathbf{R}(t_{\mathbf{x}})\mathbf{R}^T(t_{\mathbf{y}})\mathbf{K}^{-1}\mathbf{y}\,. \tag{2}$$

Here $\mathbf{K}$ is the intrinsic camera matrix, $\mathbf{R}(t)$ is the time-continuous camera rotation to be estimated, and $\sim$ denotes equality up to scale. The times $t_{\mathbf{x}}$ and $t_{\mathbf{y}}$ correspond to when the image points $\mathbf{x}$ and $\mathbf{y}$ were observed.

2.2. Translation model

In [17] the authors found that the rotation model was good for rolling-shutter rectification since rotation is the dominant cause for the distortions. This model is used pairwise on neighbouring frames, but for a global alignment between all the frames, we also take translations into account. For the translations, we approximate the scene with a fronto-parallel plane. A point $\mathbf{y}$ in one of the frames may be re-projected onto this scene plane as $\mathbf{u}$ using:

$$\mathbf{u} = \lambda\mathbf{K}^{-1}\mathbf{y} = \lambda\,(u_1\ \ u_2\ \ 1)^T\ . \tag{3}$$

Now we may add a 3D displacement $\mathbf{d} = (\Delta X\ \Delta Y\ \Delta Z)^T$, and re-project the result into the first image:

$$\mathbf{x} = \mathbf{K}(\lambda\mathbf{K}^{-1}\mathbf{y} + \mathbf{d}) = \begin{pmatrix} y_1 s + a \\ y_2 s + b \\ 1 \end{pmatrix}\ , \tag{4}$$

where $\{s, a, b\}$ are functions of the elements of $\mathbf{K}$ and $\mathbf{d}$. We may thus estimate $\{s, a, b\}$ instead of $\mathbf{d}$. Estimation of $\{s, a, b\}$ can be done efficiently from a set of corresponding points using least squares. This is the 2D equivalent of Horn's rigid motion estimation method [7].

3. Interpolation of Rotations

In order to obtain a smooth representation of the continuous rotation $\mathbf{R}(t)$, we use the cumulative quaternion splines proposed in [10]. A B-spline curve defined by knots $\mathbf{p}_k \in \mathbb{R}^n$ is evaluated as:

$$\mathbf{p}(t) = \sum_{k=0}^{K} \mathbf{p}_k B_k(t)\,. \tag{5}$$

The cumulative form of (5) is:

$$\mathbf{p}(t) = \mathbf{p}_0 \tilde{B}_0(t) + \sum_{k=1}^{K} \Delta\mathbf{p}_k \tilde{B}_k(t)\,, \text{where} \tag{6}$$

$$\Delta\mathbf{p}_k = \mathbf{p}_k - \mathbf{p}_{k-1} \quad \text{and} \quad \tilde{B}_k = \sum_{l=k}^{K} B_k\,. \tag{7}$$

In analogy with this, cumulative splines on the rotation manifold may be defined, using unit quaternion knots, $\mathbf{q}_k = (\cos\alpha_k, \hat{\mathbf{n}}_k \sin\alpha_k) \in \text{Spin}(3)$, and quaternion operations as [10]:

$$\mathbf{q}(t) = \mathbf{q}_0^{\tilde{B}_0(t)} \prod_{k=1}^{K} \exp(\boldsymbol{\omega}_k \tilde{B}_k(t)) \text{ where} \tag{8}$$

$$\boldsymbol{\omega}_k = \log(\mathbf{q}_{k-1}^* \mathbf{q}_k)\,. \tag{9}$$

Here $*$ denotes the quaternion conjugate, and for a unit quaternion $\mathbf{q}$, the logarithm is defined as:

$$\log\mathbf{q} = \log\left(\cos\alpha, \hat{\mathbf{n}}\sin\alpha\right) = (0, \alpha\hat{\mathbf{n}})\,, \tag{10}$$

and $\exp()$ is the corresponding inverse operation. In [10] B-spline kernels were used, and these define a curve by approximation. As we are interested in interpolation, we will also try replacing the B-spline kernels with the classical interpolating cubic spline (with the common choice of $\partial B/\partial t(1) = -0.5$, see e.g. [18]), as well as the following *quartic* spline:

$$B_{\text{int}}(t) = \begin{cases} -\frac{1}{2}t^4 + \frac{5}{2}|t|^3 - 3t^2 + 1 & \text{if } |t| < 1 \\ \frac{1}{2}t^4 - \frac{7}{2}|t|^3 + 9t^2 - 10|t| + 4 & \text{if } |t| \in [1,2] \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

This spline is the unique piecewise quartic that satisfies the following 10 constraints: constant sum (1dof), $B : [-2, -1, 0, 1, 2] \to [0, 0, 1, 0, 0]$ (5dof), $\partial B/\partial t$ and $\partial^2 B/\partial t^2$ continuous at $t = 1$ (2dof). $\partial B/\partial t : [0, 2] \to [0, 0]$ (2dof).

Figure 2 shows a B-spline kernel and the kernel defined in (11), and their corresponding cumulative kernels.

Note also that it is possible to move smoothly between interpolation and approximation by blending the interpolating and the approximating kernels:

$$\tilde{B}(t, \gamma) = \gamma\tilde{B}_{\text{int}}(t) + (1-\gamma)\tilde{B}_{\text{approx}}(t)\,. \tag{12}$$

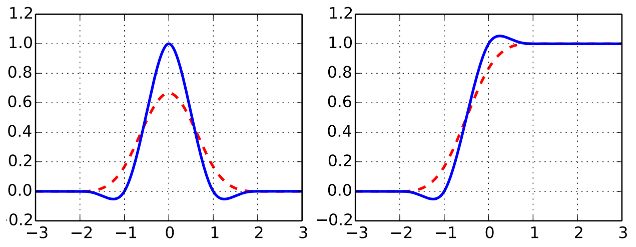

Figure 2. Kernels. Left: Interpolating quartic spline (solid blue) and B-spline (dashed red). Right: The corresponding cumulative kernels.

Here $\tilde{B}_{\text{approx}}(t)$ is the B-spline kernel, and γ is a blending parameter.

3.1. Integration

Interestingly, the tangent vectors $\boldsymbol{\omega}_k$ in (9) are closely related to angular velocities. This allows us to compute them from the gyro data $\{\mathbf{g}_k\}_0^K$, using the expression:

$$\boldsymbol{\omega}_k = (0, \Delta t(\mathbf{g}_k + \mathbf{g}_{k-1} - 2\mathbf{b})/2) \ , \tag{13}$$

where Δt is the gyro sample time, and $\mathbf{b}$ is the gyro bias vector to be estimated. The rationale for (13) is that for small angles (i.e. small Δt) it is a good approximation of trapezoidal integration on the manifold of rotations.

4. Parameter Optimisation

The use of gyro data together with a camera requires estimation of the time delay t_d between gyro samples and camera frame timestamps, as well as the three element gyro bias vector $\mathbf{b}$, see (13).

In [9] a calibration procedure that finds t_d and $\mathbf{b}$ is proposed, our approach is quite similar, but we have replaced an initial global-shutter geometric constraint with a geometry free rejection, and added a rolling-shutter geometry based rejection later on.

First the reprojection error of (2) is used to define residuals for correspondences between neighbouring frames. As we want to avoid imposing geometric constraints on the correspondences, we use cross-checking rejection on KLT-features, as suggested in [4]. We start with setting the gyro bias to the sample mean and estimate the time delay t_d using point correspondences from a few frames in the beginning of the sequence. The parameter that minimises the squared sum of the residuals is found using non-linear optimisation. After convergence of this optimisation, we obtain residuals that approximately follow a Gaussian distribution. We have found that better accuracy of the sought parameters can be obtained by removing correspondences with residuals beyond the 3σ limit at this stage. After removal of these correspondences, we optimise for both t_d and $\mathbf{b}$ using correspondences from the whole sequence until convergence.

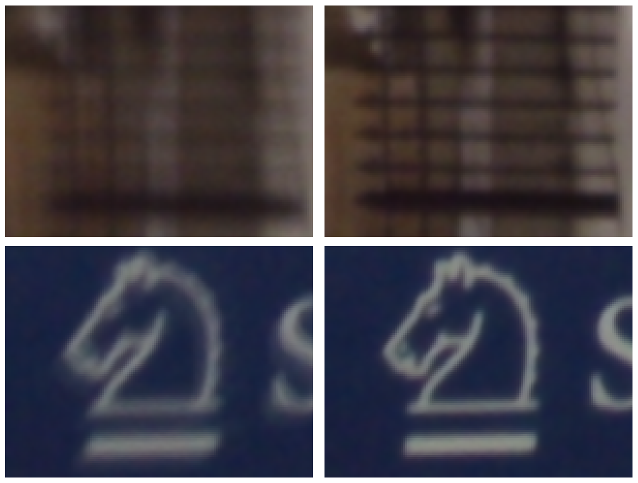

Figure 3. Zoomed in examples between global frame alignment (left) and our rolling shutter aware method (right). For full frames, see Figure 7, top left, and Figure 6, bottom left.

4.1. Robust Estimation

After t_d and $\mathbf{b}$ have been found, we can apply the rotation model (2) to all points. We do this and resample the images using forward interpolation as suggested in [4].

After image resampling, we have a fairly well aligned stack, but if the imaged scene is close to the camera we still need to apply the 3D translation model from section 2.2. In order to do this, we again run a KLT-tracker between the first frame, and each successive frame, and remove outliers using cross-checking. We then use the found correspondences to estimate the translation model (4) within a RANSAC [18] loop. For the model (4) the minimal number of sample correspondences is 2. This, and the low number of remaining outliers together mean that RANSAC usually finds the correct model after just a few trials. Once a model with a large ratio of inliers has been found, we re-estimate the model using all inliers.

In the final result, we want to avoid any unnecessary blurring caused by resampling the images twice. The final correction is thus obtained by applying both the rotation model and the translation model to the original image coordinates, and then resampling the original frames using forward interpolation.

4.2. Algorithm Bottlenecks

The proposed algorithm is quite efficient. The current implementation is in Python, and runs on a PC, for ease of analysis. Currently, the most expensive part of the algorithm is image resampling and saving to disk, followed by optimisation, and feature tracking. Most of the time is currently spent on saving to disk and image re-sampling, but if resampling was to be done on the smartphone GPU, the cost

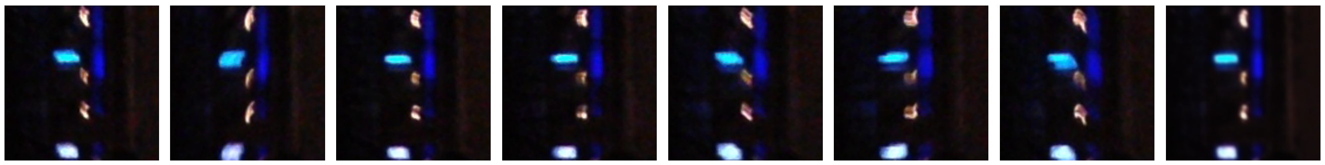
Figure 4. Columns 1-7: Example frames from a hand-held sequence showing different blurs. Right: Our result. (Best viewed electronically)

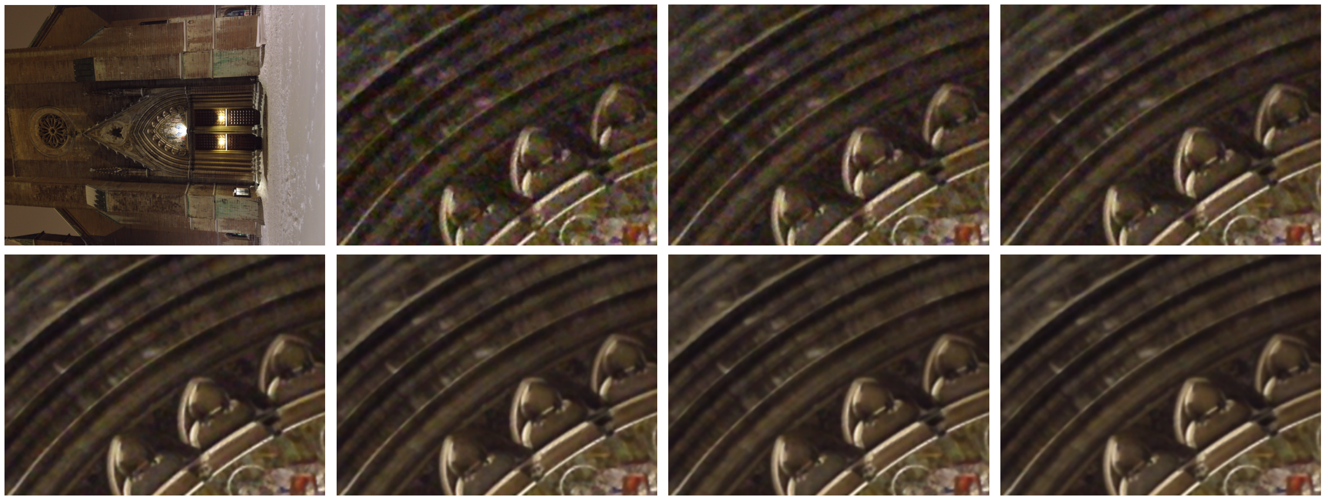
Figure 5. Scene captured using a physical tripod. Left to right, top to bottom: denoised image using 64 frames, zoomed in detail using 1, 2, 4, 8, 16, 32 and 64 frames respectively. (Best viewed electronically)

of these steps would be negligible. Thus, an efficient implementation on a smartphone should be straight forward.

5. Experiments

We have tested our method on many real-life sequences and compare our results with the following methods: (1) the single frame denoising implemented in Photoshop CS 5.1[1], (2) direct averaging of the unaligned frames in the stack, (3) global frame alignment (i.e. without rolling shutter correction). In the experiments, our method uses 32 frames, and B-spline kernels, unless stated otherwise.

When capturing images in low light we usually get both motion blur and image noise. Since most of the motion blur is from rotation, the blur kernel is non-uniform, both across the frame [21] and temporally. In Figure 4 we give an example of how different it may look like across a sequence of frames from a hand-held sequence.

If we assume a rigid scene we could use as many frames as we want to obtain the final result, but since this is not always the case we have to trade the capture duration and the output noise level. In Figure 5 the noise level for stacks with increasing number of frames is shown. In this particular example, improvements beyond 16 frames are difficult to see.

[1]Note that Photoshop CC also has a Shake Reduction feature, which has not been tested here.

5.1. Data collection

We have implemented an app for iOS that logs timestamped full resolution frames, as well as gyro sensor data at 100 Hz. The obtained frame-rate is a function of the bus-speed, the chosen video-quality, and the computational power of the device. We have configured the app to record using the JPEG encoder, with quality set to 85%. This results in a recording speed of about 9 Hz on iPhone 4s, 14 Hz on iPhone 5, and 30 Hz on iPhone 5s. For the same amount of denoising, the 5s thus has a shorter stacking time, due to its superior light collection efficiency, see (1).

The five stacks used in this experiment have been made available in a public dataset[2]. This includes full resolution input images, frame timestamps, and logs from the builtin gyroscope.

5.2. Comparative Experiments

Here we compare our results with a single frame from the stack, a denoising of this frame using Photoshop CS 5.1, and the average of the non-aligned frames in the sequence, see Figures 6 and 7. The Photoshop denoising was set to standard settings except "strength" and "preserve details" which were changed to 10 and 10% respectively.

It is also interesting to see how our method compares to a global alignment of the frames. In order to do this we

[2]Dataset: `http://www.cvl.isy.liu.se/research/datasets/stacking-dataset/`

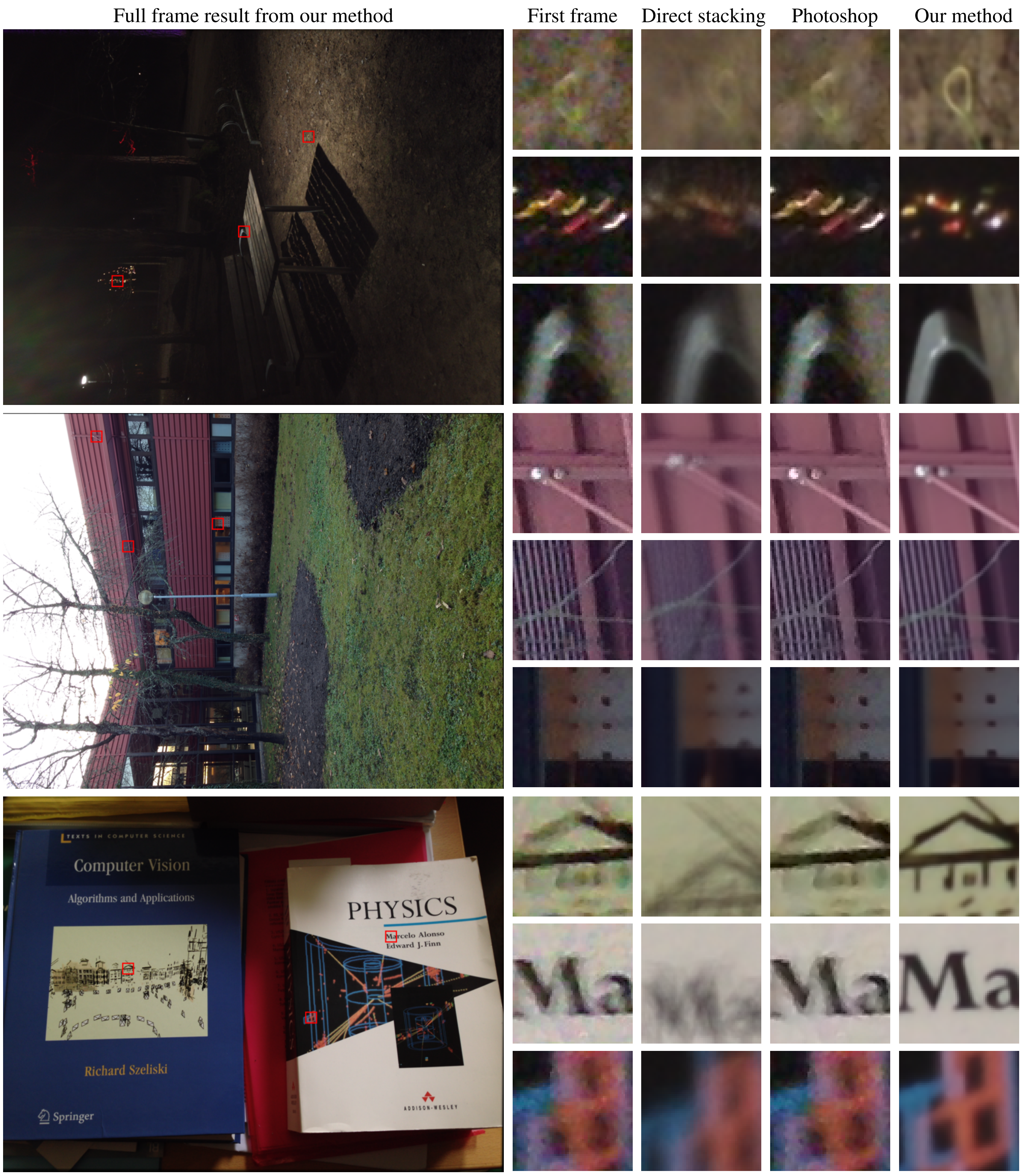

Figure 6. Results for Table (iPhone 4s), Grass (iPhone 5), and Books (iPhone 5) datasets. Left: Full frames after stacking with our method. Right columns: first frame in sequence, stacking of original frames, denoised first frame using Photoshop and our results. (Best viewed electronically)

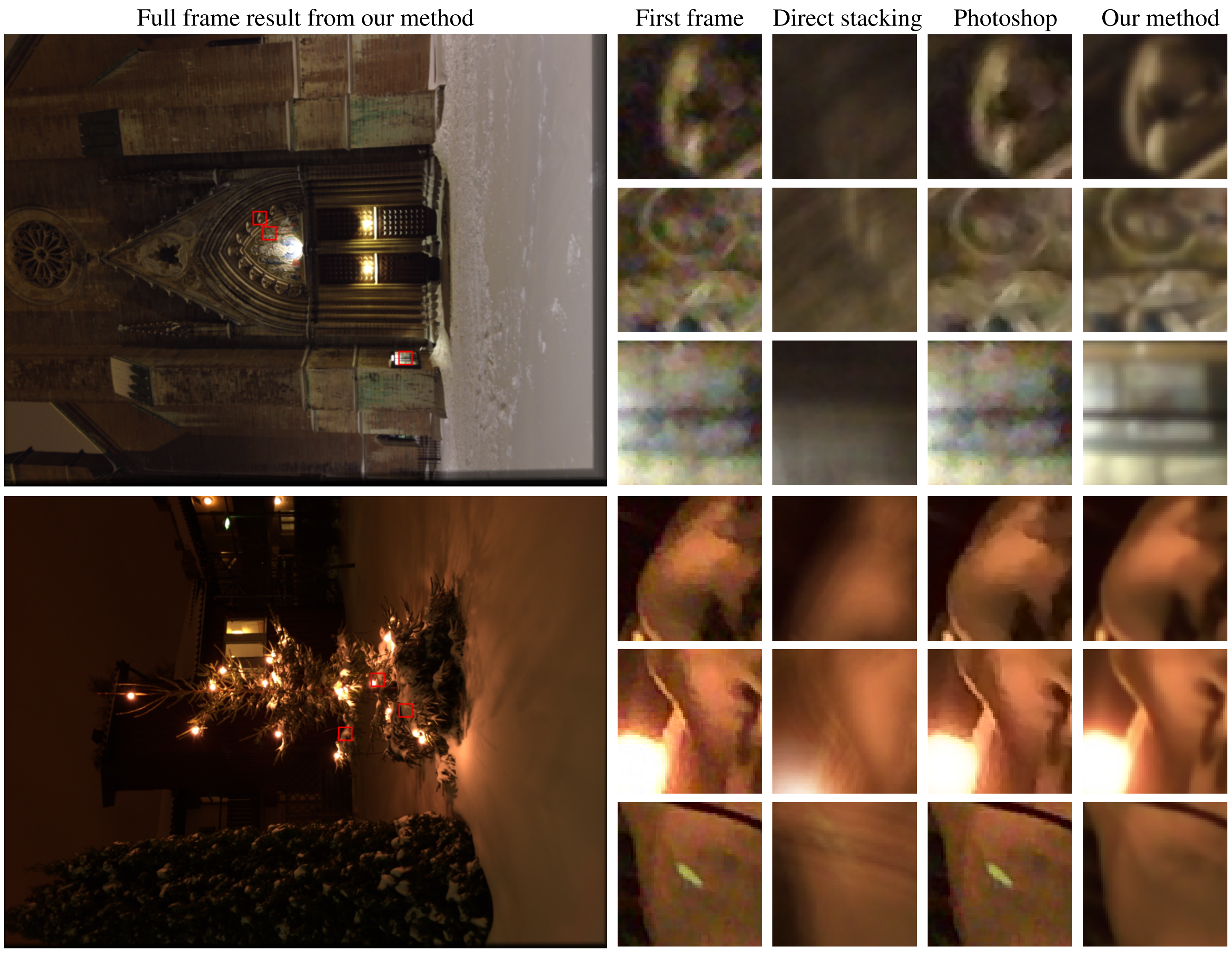

Figure 7. Results for Church (iPhone 4s) and Tree (iPhone 5s) datasets. Left: Full frame after stacking with our method. Right columns: first frame in sequence, stacking of original frames, denoised first frame using Photoshop and our results. (Best viewed electronically)

used our estimated motion and applied a global rotation and translation on each frame. Please note that this motion has been estimated taking rolling shutter into account and that the first row of the global alignment will thus be the same as in our method. Figure 3 shows how the global alignment gets worse further down the image, whereas the proposed method has consistent performance at all image rows. As can be seen in the detail subplots, our algorithm successfully averages out the noise, while preserving structural details. An interesting observation is that our algorithm often manages to average out lens flares, (see e.g. Figure 7, bottom), as these move around quite a bit when shooting handheld photos. In the case of lens-flares this is a desired behaviour, but of course actual scene objects that move during the $1 - 2$ sec exposure will also be averaged out.

5.3. Limitations

The proposed method, in essence, implements the behaviour of a tripod, and as such the final result is sensitive to moving objects in the scene. For small objects, the translation estimation will lock on to the background scene, and the moving objects will be smeared just like on a tripod. For large objects, on the other hand, the translation estimation will tend to lock on to the object instead. If the object satisfies the assumption of a fronto-parallel plane, see section 2.2, the result will be a sharp object, and a smeared background. In general however the result is unpredictable. Note however that other stacking functions than the frame average that we currently use, may to some extent remedy these limitations.

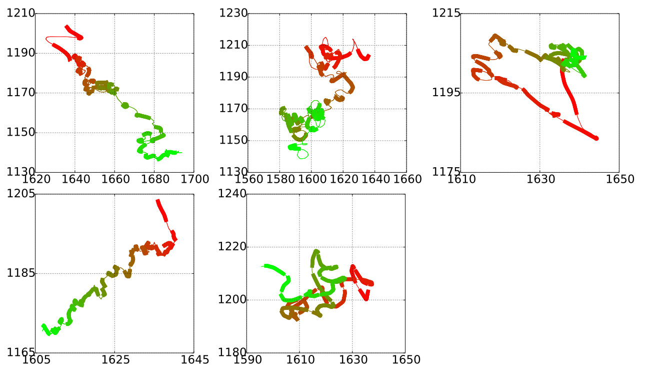
Figure 8. Trace of central pixel for the five stacks used in our experiments. Colours indicate time, ranging from red to green, and thick segments indicate the 32 individual exposures. Left to right, top to bottom: Table (4s), Grass (5), Books (5), Church (4s), Tree (5s)

5.4. Estimated PSFs from Gyroscope data

We have used an externally mounted L3G4200D gyro to record the device motion at 800Hz, as illustrated in Figure 1. Using the built-in gyro recording at 100Hz in the three iPhones (4s,5,5s), we obtain similar pixel traces, shown in Figure 8. By comparing the curves in Figure 1 and Figure 8, we see that the curves are similar, and thus conclude that the 100Hz sampling is sufficient.

In Figure 9 we have plotted iso-contours of the effective PSF for the central pixel in each of the datasets. The contour levels are set to $1/255$, $2/255 \ldots 32/255$. This means that beyond the first iso contour, the central pixel will not be influenced beyond the 8-bit quantization level. Beyond the second contour, a change of more than 128 in pixel value is required to influence the blurred pixel value, and so on.

It is interesting to relate the PSFs in Figure 9 to the imaging situation. The Grass sequence was imaged with elbows resting on a ledge, and consequently it has the smallest PSF. The Church and Tree sequences were recorded in cold weather, and consequently they have slightly more handshake. The Table, Books, and the 800Hz recording in Figure 1 were all recorded in warmer conditions, and consequently have better concentrated PSFs.

5.5. Quantitative Experiments

For quantitative evaluation of the stacking result, we use the standard deviation in time across a stack of frames. This measure is then averaged across all pixels to obtain a scalar measure. If we denote the c-th colour band of RGB frame

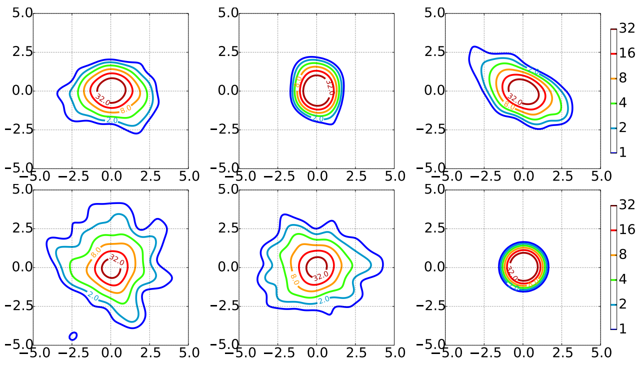
Figure 9. Iso contours of the effective PSF for central pixel. PSF values are scaled by 255, see text for details. Left to right, top to bottom: Table (4s), Grass (5), Books (5), Church (4s), Tree (5s), Gaussian with $\sigma = 0.5$.

k in a stack by $I_{k,c}(\mathbf{x})$, the measure is computed as:

$$\sigma_{\text{avg}} = \frac{1}{3|\Omega|} \sum_{\mathbf{x}\in\Omega} \sum_{c=1}^{3} \sqrt{\frac{1}{K} \sum_{k=1}^{K} (I_{k,c}(\mathbf{x}) - I_{\text{avg},c}(\mathbf{x}))^2}\,, \tag{14}$$

$$\text{where} \quad I_{\text{avg},c}(\mathbf{x}) = \frac{1}{K} \sum_{k=1}^{K} I_{k,c}(\mathbf{x})\,. \tag{15}$$

Here Ω is the set of image coordinates in the frames, and $|\Omega|$ is the set size.

In Table 1 we use the measure (14) to compare different stacking approaches. Here **Global** refers to the global-shutter based frame alignment, also used in Figure 3. The other methods (**Slerp**, **Cubic**, **Quartic**, and **B-spline**) are versions of our rolling-shutter based algorithm, with different interpolation kernels. As can be seen in Table 1, all rolling-shutter based stacking approaches are significantly better than the global-shutter based alignment. We can also see that the **B-spline** kernel is slightly better than than the other approaches. The reason for this is probably that its low-pass characteristic results in a denoising of the gyro signal. It is also interesting to note that **Slerp** performs surprisingly well. This may be caused by it being linear, just like the trapezoid integration in (13).

We have also tried blending the **Quartic** and the **B-spline** kernels according to (12), and then got the best results for a pure **B-spline** kernel. As the performance differences are quite small, these results may however not be significant, and are thus excluded from the Table.

6. Concluding Remarks

We have introduced an algorithm for accurate stacking of full resolution frames on a smartphone. This algorithm enables hand-held capture of low-noise images in low-light conditions, and thus implements a virtual tripod. This is

Dataset		Table	Grass	Books	Church	Tree
iPhone Device		4s	5	5	4s	5s
Global	σ_{avg}	6.17	6.83	7.55	8.02	5.23
Slerp	σ_{avg}	6.00	4.80	6.43	7.01	**4.31**
	residual	0.707	**0.611**	1.11	1.10	1.04
Cubic	σ_{avg}	6.00	4.77	6.42	7.02	4.31
	residual	0.728	0.612	1.12	1.16	1.11
Quartic	σ_{avg}	6.00	4.78	6.41	7.02	4.31
	residual	0.721	0.612	1.11	1.13	1.09
B-spline	σ_{avg}	**6.00**	**4.77**	**6.41**	**7.00**	4.31
	residual	**0.692**	0.612	**1.09**	**1.05**	**0.983**

Table 1. Quantitative results for different versions of our method, on the five datasets "Table", "Grass", "Books", "Church", and "Tree". The measure σ_{avg} is defined in (14), and "residual" is the mean squared residual of the reprojection error on tracked features. Best results in each column are shown in boldface. See Figures 6 and 7 for images of the different datasets.

accomplished using high accuracy motion estimation using logged gyro sensor data and correspondences from image feature tracking. In the experiments we demonstrate that the use of cumulative quaternion splines for motion interpolation results in a more accurate stacking than currently used stacking approaches that implicitly assume a global shutter. We also demonstrate that, while Photoshop style denoising works well in some situations, our algorithm consistently delivers a sharp, low-noise output.

The proposed motion estimation could also be used in other applications where accurate frame alignment is needed, such as flash-no-flash photography, and HDR imaging using exposure brackets.

When stacking is used in astrophotography, it is common to apply deconvolution to the stacking result, e.g. using Richardson-Lucy (RL) [16, 13]. This could also be done here to obtain a sharper final image. Since the PSFs in each frame are different, another possibility is to apply RL to the individual frames, *before* stacking. This will come at a higher computational cost, but as ringing artifacts tend to depend on both image structure and image smear, this may improve the output quality.

In the paper we have only investigated frame combination using direct averaging. In future research we plan to investigate how this compares to other commonly used approaches, such as temporal median and bilateral filtering. It would also be interesting to investigate criteria for stopping frame acquisition automatically when sufficient data is available to average out the noise. Finally, moving the entire algorithm onto a smartphone will also be tested.

References

[1] K. Brasch. The origin of stacking. *Sky and Telescope*, March 2014.

[2] Sony RX100 review. www.dpreview.com, accessed December 18, 2013.

[3] E. Eisemann and F. Durand. Flash photography enhancement via intrinsic relighting. *ACM Trans. Graph.*, 23(3):673–678, Aug. 2004.

[4] P.-E. Forssén and E. Ringaby. Rectifying rolling shutter video from hand-held devices. In *CVPR10*, San Francisco, USA, June 2010. IEEE Computer Society.

[5] M. Grundmann, V. Kwatra, D. Castro, and I. Essa. Effective calibration free rolling shutter removal. *IEEE ICCP*, 2012.

[6] G. Hanning, N. Forslöw, P.-E. Forssén, E. Ringaby, D. Törnqvist, and J. Callmer. Stabilizing cell phone video using inertial measurement sensors. In *2nd IEEE IWMV*, 2011.

[7] B. K. P. Horn. Solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am.*, 4:629–642, April 1987.

[8] N. Joshi, S.-B. Kang, L. Zitnick, and R. Szeliski. Image deblurring using inertial measurement sensors. In *SIGGRAPH'10*, 2010.

[9] A. Karpenko, D. Jacobs, J. Baek, and M. Levoy. Digital video stabilization and rolling shutter correction using gyroscopes. Technical Report CSTR 2011-03, Stanford University Computer Science, September 2011.

[10] M.-J. Kim, M.-S. Kim, and S. Y. Shin. A general construction scheme for unit quaternion curves with simple high order derivatives. In *SIGGRAPH'95*, pages 369–376, 1995.

[11] C. Liu and W. Freeman. A high-quality video denoising algorithm based on reliable motion estimation. In *European Conference on Computer Vision (ECCV10)*, 2010.

[12] S. Lovegrove, A. Patron-Perez, and G. Sibley. Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras. In *British Machine Vision Conference (BMVC'13)*, 2013.

[13] L. B. Lucy. An iterative technique for the rectification of observed distributions. *Astron. J.*, 79:745–754, 1974.

[14] G. Petschnigg, M. Agrawala, H. Hoppe, R. Szeliski, M. Cohen, and K. Toyama. Digital photography with flash and no-flash image pairs. In *Proceedings of SIGGRAPH'04*, 2004.

[15] T. Portz, L. Zhang, and H. Jiang. High-quality video denoising for motion-based exposure control. In *2nd IWMV*, 2011.

[16] W. H. Richardson. Bayesian-based iterative method of image restoration. *J. Opt. Soc. Am.*, pages 55–59, 1972.

[17] E. Ringaby and P.-E. Forssén. Efficient video rectification and stabilisation for cell-phones. *IJCV*, 96(3), 2012.

[18] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer Verlag, 2011.

[19] O. Šindelář and F. Šroubek. Image deblurring in smartphone devices using built-in inertial measurement sensors. *Journal of Electronic Imaging*, 22(1), 2013.

[20] J. Weickert. *Anisotropic Diffusion in Image Processing*. ECMI Series, Teubner Verlag, Stuttgart, Germany, 1998.

[21] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce. Non-uniform deblurring for shaken images. In *IEEE CVPR*. IEEE Computer Society, June 2010.

[22] L. Yuan, J. Sun, L. Quan, and H.-Y. Shum. Image deblurring with blurred/noisy image pairs. In *SIGGRAPH'07*, 2007.

[23] H. Zhang and D. Wifp. Non-uniform camera shake removal using a spatially-adaptive sparse penalty. In *NIPS13*, 2013.