Rolling Shutter Bundle Adjustment

Johan Hedborg Per-Erik Forssén Michael Felsberg Erik Ringaby
Computer Vision Laboratory, Department of Electrical Engineering Linköping University, Sweden

Abstract

This paper introduces a bundle adjustment (BA) method that obtains accurate structure and motion from rolling shutter (RS) video sequences: RSBA. When a classical BA algorithm processes a rolling shutter video, the resultant camera trajectory is brittle, and complete failures are not uncommon. We exploit the temporal continuity of the camera motion to define residuals of image point trajectories with respect to the camera trajectory. We compare the camera trajectories from RSBA to those from classical BA, and from classical BA on rectified videos. The comparisons are done on real video sequences from an iPhone 4, with ground truth obtained from a global shutter camera, rigidly mounted to the iPhone 4. Compared to classical BA, the rolling shutter model requires just six extra parameters. It also degrades the sparsity of the system Jacobian slightly, but as we demonstrate, the increase in computation time is moderate. Decisive advantages are that RSBA succeeds in cases where competing methods diverge, and consistently produces more accurate results.

1. Introduction

Structure from motion (SfM) is one of the success stories in computer vision [11]. SfM is now routinely used to add visual effects to video, e.g. in the movie industry, and it has been successfully used to build 3D models from both photo collections, and from video [24]. Another technology that uses SfM as its back-end is augmented reality [16].

An overwhelming majority of image sensors sold today are of CMOS type: nearly all mobile video recording devices, and most compact cameras have them. In contrast to the classical CCD sensors, which have global sensor readout, the image rows of CMOS sensors are read out in rapid succession over a *readout time* of 10-60 msec [28]. In addition, modern video recording devices come without a mechanical shutter, and instead reset the sensor elements electronically. These two effects combined constitute what is known as an *electronic rolling shutter*, and lead to a *rolling shutter* (RS) camera model [10].

Most work on SfM is based on the global shutter camera model [29, 11]. When used on rolling shutter cameras these algorithms become brittle, e.g. Liu et al. [18] demonstrate several cases where the Voodoo tracker[1] fails, and similarly Hedborg *et al*. [12] demonstrate failure of the SBA package of Lourakis and Agyros [19] under rolling shutter.

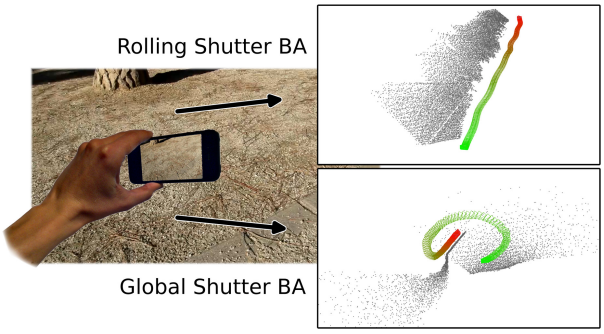


Figure 1. If classical structure from motion is applied to rolling shutter video, the result is unpredictable, whereas **RSBA** is stable. These results are from sequence #19.

1.1. Related work

Bundle adjustment (BA) is a collective name for techniques that refine an initial estimate of structure and camera motion, by minimising the reprojection errors over all images [29]. It has been shown that it is possible to use BA solvers even in real time applications to improve SfM estimation [16, 6]. Other works have studied techniques for improving the robustness of BA [22]. Recent progress has been made in terms of stability and speed, especially for large scale problems where several thousands cameras and millions of points are refined and BA can now be used to solve even city scale problems [9, 1, 13].

Despite the prevalence of rolling shutter cameras, systems that model rolling shutter cameras are rare in the literature, and all previous work has modelled special cases. In contrast, we present a system where the continuous six degree-of-freedom camera trajectory is modelled under rolling shutter geometry.

In an early study by Geyer *et al*. [10], rolling-shutter SfM is estimated on synthetic data for fronto-parallel motions,

[1] http://www.digilab.uni-hannover.de/

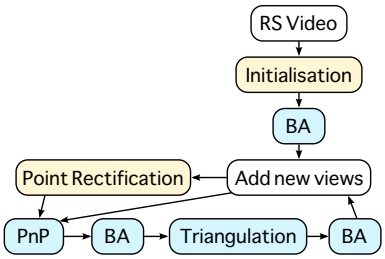


Figure 2. Flow-chart of the sequential SfM pipeline. Yellow boxes were added/modified by [12]. In this paper, we skip the point rectification step, and instead make rolling-shutter versions of the blue boxes. In [2, 16], only the PnP box was modified, and special requirements on the initialisation were required.

and with a linearised screw motion model. Ego-motion under known structure and rolling shutter cameras is studied in [3]. A related study considers structure and motion on a stereo rig where one of the cameras has a rolling shutter [4].

Ait-Aider *et al*. [2] solved the *perspective-n-point* (PnP) [7] problem for rolling-shutter cameras where the camera pose, and linear camera motion is estimated across one frame only. Another PnP solution is the PTAM port to iPhone 3G by Klein *et al*. [16]. As both of these solutions use 2D-3D correspondences, they require an initially known 3D structure. In the PTAM case the initial 3D structure is found by requiring that the start of the sequence images a planar scene. As the initial 3D structure is assumed to be correct, the solution can easily deteriorate over time.

Another recent line of work is to first rectify the frames, and then apply the classical global shutter SfM pipeline [12]. While this has been demonstrated to work in several cases, the accuracy of the reconstruction is critically dependent on the initial rectification, and any model errors in the rectification will also propagate to the final solution.

Figure 2 is an overview of the SfM pipeline. First an initial structure and motion estimate is found using techniques described in section 4. This is first bundle-adjusted, and then new views are added in sequential fashion as shown in the cycle at the bottom of the flow-chart. The **contributions of this paper** consist in making rolling-shutter aware versions of the blue boxes, in particular we present **the first rolling shutter bundle adjustment** method.

2. Bundle Adjustment

With Bundle Adjustment we refer to the process of refining the complete set of camera parameters and 3D point positions such that the error between the observed image points and the projection of the 3D points is minimised (aka. the *reprojection error*). The most common approach to estimate these parameters is to pose it as a non-linear least squares problem and solve it with the Levenberg-Marquardt algorithm [17].

The distance metric between the reprojected point and the observed point can either be the L_2 norm, or differentiable functions thereof, e.g. based on the Cauchy error distribution [6]. In this paper we follow the example of [19, 13] and use the L_2 norm error, but this can easily be modified to use another norm if needed.

2.1. Levenberg-Marquardt Algorithm

The Levenberg-Marquardt algorithm is an iterative method for minimising the quadratic norm of a vector valued residual function $\mathbf{r}(\mathbf{x})$

$$\min_{\mathbf{x}} \; 1/2 \, ||\mathbf{r}(\mathbf{x})||^2 \, . \tag{1}$$

Each iteration, $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}$, solves a linear problem, based on the Taylor expansion of $\mathbf{r}$ in $\mathbf{x}_k$ [21]. The update $\Delta\mathbf{x}$ is determined from the (damped) normal equations

$$(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \lambda \, \mathrm{diag}(\mathbf{J}^{\mathrm{T}}\mathbf{J}))\Delta\mathbf{x} = -\mathbf{J}^{\mathrm{T}}\mathbf{r}(\mathbf{x}_k) \, , \tag{2}$$

where $\lambda > 0$ is a damping parameter, and $\mathbf{J} = \mathbf{J}(\mathbf{x}_k)$ is the Jacobian $\mathbf{J} = \frac{\partial(r_1, r_2, \ldots)}{\partial(x_1, x_2, \ldots)}$ evaluated at point $\mathbf{x}_k$. The term $\mathbf{J}^{\mathrm{T}}\mathbf{J}$ is an approximation of the Hessian of $\mathbf{r}(\mathbf{x})$. In order to make the system better conditioned, we also scale our system with a Jacobi preconditioner, as described in [1, 13].

2.2. Solving the normal equations

The linear system (2) grows large even for moderate problems with a few hundred cameras. We use calibrated cameras, which means 6 parameters for each camera pose, and use 3 parameters for each 3D point. E.g. 200 cameras and 10K 3D points gives a 31K×31K matrix, which is not feasible to solve on most PCs, mainly due to memory usage.

Fortunately the 3D points and the cameras can be seen as independent from each other, leading to sparse Jacobian and approximate Hessian matrices. This independence is weaker in the case of a rolling shutter system as we will see later. An efficient handling of the sparsity is the key to solve (2) in an efficient way and this is what distinguishes a BA solver from a generic numerical solver.

The sparsity is typically exploited by applying the Schur complement trick [29], which in a sense is a normal block Gaussian elimination. The parameter vector consists of camera parameters, $\mathbf{c}$, and 3D model points, $\mathbf{m}$, according to $\mathbf{x}^{\mathrm{T}} = [\mathbf{c}^{\mathrm{T}} \, \mathbf{m}^{\mathrm{T}}]$. The Jacobian is split accordingly into $\mathbf{J} = [\mathbf{J}_c \; \mathbf{J}_m] = [\frac{\partial(r_1, r_2, \ldots)}{\partial(c_1, c_2, \ldots)} \; \frac{\partial(r_1, r_2, \ldots)}{\partial(m_1, m_2, \ldots)}]$, resulting in the approximate Hessian:

$$\begin{bmatrix} \mathbf{J}_c^{\mathrm{T}}\mathbf{J}_c & \mathbf{J}_c^{\mathrm{T}}\mathbf{J}_m \\ \mathbf{J}_m^{\mathrm{T}}\mathbf{J}_c & \mathbf{J}_m^{\mathrm{T}}\mathbf{J}_m \end{bmatrix} + \lambda \, \mathrm{diag} \begin{bmatrix} \mathbf{J}_c^{\mathrm{T}}\mathbf{J}_c & 0 \\ 0 & \mathbf{J}_m^{\mathrm{T}}\mathbf{J}_m \end{bmatrix} = \begin{bmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^{\mathrm{T}} & \mathbf{V} \end{bmatrix} . \tag{3}$$

The normal equations (2) now read

$$\begin{bmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^{\mathrm{T}} & \mathbf{V} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{c} \\ \Delta\mathbf{m} \end{bmatrix} = - \begin{bmatrix} \mathbf{J}_c^{\mathrm{T}} \\ \mathbf{J}_m^{\mathrm{T}} \end{bmatrix} \mathbf{r} \, . \tag{4}$$

The camera parameter update can now be computed separately by elimination

$$\left(\mathbf{U} - \mathbf{W}\mathbf{V}^{-1}\mathbf{W}^{\mathrm{T}}\right)\Delta\mathbf{c} = \left(\mathbf{W}\mathbf{V}^{-1}\mathbf{J}_m^{\mathrm{T}} - \mathbf{J}_c^{\mathrm{T}}\right)\mathbf{r}\,. \quad (5)$$

The method of choice for solving the linear system (5) is a Cholesky factorization due to the symmetry of the coefficient matrix (which is the Schur complement). How to do this efficiently will be described in section 3.3.

Once we have the camera update, the update for the 3D points is obtained as:

$$\Delta\mathbf{m} = -\mathbf{V}^{-1}(\mathbf{J}_m^{\mathrm{T}}\mathbf{r} + \mathbf{W}^{\mathrm{T}}\Delta\mathbf{c})\,. \quad (6)$$

Note that $\mathbf{V}$ is 3×3 block diagonal, and this step is thus very inexpensive. There is also a second sparsity structure in the Schur complement (due to points not being visible in all cameras). This becomes relevant when dealing with large problems as noted in [1, 6].

3. Rolling Shutter Bundle Adjustment

We present a Bundle Adjustment solver for rolling shutter cameras. We have chosen to look at video sequences because this case is more tightly coupled than the image collection case. In single frame rolling shutter models, we get six [16], or more [2] extra camera parameters *per frame*. In our approach, we interpolate between camera poses for the first row of each frame, and instead get a total of six extra parameters *for the entire sequence*.

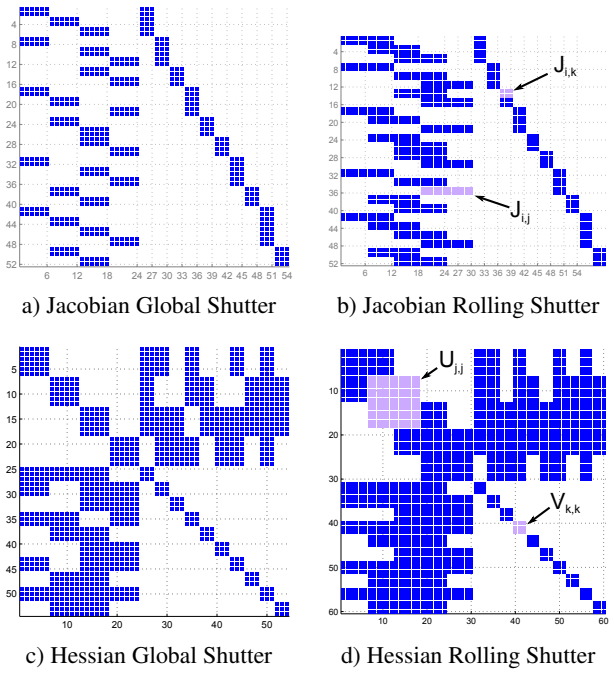
3.1. Camera Model

In a rolling shutter camera, image rows are captured at different time instances in sequential order. In the general case this leads to different cameras poses for each row. Trying to solve for all of these would lead to a heavily underdetermined system due to too few measurements. This problem can be handled by only estimating a subset of the poses, and representing the remaining ones using interpolation. Many interpolation schemes can be used here but the general rule is that if we increase the complexity of the interpolation, we also increase the camera dependencies and thus reduce the sparsity of the system Jacobian.

The interpolation chosen here follows the one proposed in [8], using SLERP interpolation for the rotation [27] and linear interpolation for the translation. We place key rotations $\mathbf{R}_j$ and translations $\mathbf{t}_j$ at the first row of each frame j. The rolling shutter camera model for row y in frame j reads

$$\mathbf{C}_j(y) = \mathbf{R}_{j,j+1}^{T}(y)[\,\mathbf{I}\mid\, -\mathbf{t}_{j,j+1}(y)] \quad (7)$$

where $\mathbf{R}_{j,j+1}(y)$ is the SLERP interpolated rotation between $\mathbf{R}_j$ and $\mathbf{R}_{j+1}$, and similarly $\mathbf{t}_{j,j+1}(y)$ is the interpolated translation. The method works for any number of key rotations and translations per frame but we have chosen to use just one per frame. In practise, a six parameter *key pose vector* $\mathbf{c}_j$ is used to represent a key pose $\{\mathbf{R}_j, \mathbf{t}_j\}$.



a) Jacobian Global Shutter b) Jacobian Rolling Shutter

c) Hessian Global Shutter d) Hessian Rolling Shutter

Figure 3. Block structure for the Jacobian and the approximate Hessian for the case of Global Shutter and Rolling Shutter. This is a small problem with 4 cameras and 10 points.

3.2. Structural Changes

The primary structure for a global shutter Bundle Adjustment Jacobian and approximate Hessian can be seen in the left column of figure 3. This example uses a system of calibrated cameras with indices $j \in [1\ldots4] = \mathcal{J}$, and 3D points with indices $k \in [1\ldots10]$ that generate image projections with indices $i \in [1\ldots26] = \mathcal{I}$. Each 2x6 block (indexed by i and j) in figure 3a, left, consists of the two residuals in an image w.r.t. the 6 extrinsic parameters of a camera. Each $2I$x3 block (indexed by i and k) in the right half of 3a, comes from a 3D point being seen in I images.

The structural differences between the global shutter and the rolling shutter Jacobians are minor. The added dependency between one camera pose and the next doubles the width of the camera sub-Jacobians, see figure 3b. The Jacobian has also grown with 6 columns, as an extra camera pose has been added just beyond the last camera.

The matrix products of the Jacobians with their respective transposes (i.e. the approximate Hessians) differ in the two cases as illustrated in figures 3c and d.

3.3. An Efficient Implicit Solution

The update step that finds $\Delta\mathbf{c}$ and $\Delta\mathbf{m}$, see (4), can be solved implicitly using only the memory size of the matrix $\mathbf{U}$, if $\mathbf{J}_c$ and $\mathbf{J}_m$ are stored in a sparse format. As in classical bundle adjustment the whole process is linear in the number of points, and quadratic in the number of cameras. First we

solve for the camera update (5):

- Let $\mathcal{I}_k$ denote the index set of the image plane residuals of 3D point $\mathbf{m}_k$ (cf. figure 3b), and let $\mathcal{J}_k$ denote the index set of the corresponding cameras $\mathbf{C}_j, j \in \mathcal{J}_k$. Let $\mathbf{J}_{i,j}$ be the 2x12 sub-Jacobian that relates $\mathbf{C}_j$ and the residual $i = i(j,k)$ for the 3D point $\mathbf{m}_k$, as illustrated in figure 3b. The Jacobian $\mathbf{J}_c$ is formed by the union of all $\mathbf{J}_{i,j}$, for $i \in \mathcal{I}, j \in \mathcal{J}$.
- Compute $\mathbf{U}$ sequentially: First initialise $\mathbf{U} = \mathbf{0}$, then, for each 3D point $\mathbf{m}_k$ do :
  - For all cameras $\mathbf{C}_j$, $j \in \mathcal{J}_k$, $\mathbf{U}$ is updated according to $\mathbf{U}_{j,j} \leftarrow \mathbf{U}_{j,j} + \mathbf{J}_{i,j}^{\mathrm{T}}\mathbf{J}_{i,j}$, where $i = i(j,k)$ and $\mathbf{U}_{j,j}$ is the 12x12 sub-matrix of $\mathbf{U}$ corresponding to camera $\mathbf{C}_j$, see figure 3d.
- To apply the regularization, we can now simply multiply each diagonal element of $\mathbf{U}$ with $1 + \lambda$.
- Compute $\mathbf{b} = -\mathbf{J}_c^{\mathrm{T}}\mathbf{r}$ sequentially. First set $\mathbf{b} = \mathbf{0}$, then, for each 3D point $\mathbf{m}_k$ do
  - For all cameras $\mathbf{C}_j$, $j \in \mathcal{J}_k$ and their corresponding residuals $\mathbf{r}_i$, $i = i(j,k)$, do $\mathbf{b}_j \leftarrow \mathbf{b}_j - \mathbf{J}_{i,j}^T\mathbf{r}_i$, where $\mathbf{b}_j$ is a 12x1 sub-matrix of $\mathbf{b}$, corresponding to $\mathbf{C}_j$.
- Compute $\mathbf{S} = \mathbf{U} - \mathbf{W}\mathbf{V}^{-1}\mathbf{W}^T$ and $\mathbf{b} \leftarrow \mathbf{b} + \mathbf{W}\mathbf{V}^{-1}\mathbf{J}_m^{\mathrm{T}}\mathbf{r}$. Initialize $\mathbf{S} = \mathbf{U}$. For each 3D point $\mathbf{m}_k$ do :
  - Let $\mathbf{J}_{i,k}$ be the 2x3 sub-Jacobian for 3D point index k and image point index $i \in \mathcal{I}_k$. The Jacobian $\mathbf{J}_m$ is formed by the union of all $\mathbf{J}_{i,k}$, for $i \in \mathcal{I}$, and $j \in \mathcal{J}$.
  - Construct the 3x3 matrix $\mathbf{V}_{k,k} = \sum_{i \in \mathcal{I}_k} \mathbf{J}_{i,k}^T\mathbf{J}_{i,k}$.
  - For all combinations of cameras $(\mathbf{C}_{j_1}, \mathbf{C}_{j_2})$, where $j_1, j_2 \in \mathcal{J}_k$ (accordingly $i_1 = i(j_1,k)$ and $i_2 = i(j_2,k)$), update sub-matrices of $\mathbf{S}$ as $\mathbf{S}_{j_1,j_2} \leftarrow \mathbf{S}_{j_1,j_2} - \mathbf{J}_{i_1,j_1}^{\mathrm{T}}\mathbf{J}_{i_1,k}\mathbf{V}_{k,k}^{-1}\mathbf{J}_{i_2,k}^{\mathrm{T}}\mathbf{J}_{i_2,j_2}$ and $\mathbf{b}_{j_1} \leftarrow \mathbf{b}_{j_1} + \mathbf{J}_{i_1,j_1}^{\mathrm{T}}\mathbf{J}_{i_1,k}\mathbf{V}_{k,k}^{-1}\mathbf{J}_{i_2,k}^{\mathrm{T}}\mathbf{r}_{i_2}$.
- Finally, the update step for the cameras is completed by solving the symmetric linear system $\mathbf{S}\Delta\mathbf{c} = \mathbf{b}$.

The points update (6) is computationally of low complexity and is implemented in the following way:

- For each 3D point $\mathbf{m}_k$ do :
  - Compute: $\mathbf{a}_i = \mathbf{r}_i + \mathbf{J}_{i,j}\Delta\mathbf{c}_j$ for all $i = i(j,k) \in \mathcal{I}_k$, (and thus $j \in \mathcal{J}_k$).
  - Compute the 3D point update as: $\Delta\mathbf{m}_k = -\mathbf{V}_{k,k}^{-1} \sum_{i \in \mathcal{I}_k} \mathbf{J}_{i,k}^{\mathrm{T}}\mathbf{a}_i$, where

Further efficiency can be gained by using symmetry to avoid repeating some computations twice. For instance, a more efficient (but less readable) implementation can be found by reordering the cameras, and using BLAS3[2] [13].

[2]BLAS3 is a library of matrix-matrix operations.

3.4. Jacobian Calculations

Each sub-Jacobian $\mathbf{J}_{i,j}$ contains the derivatives of $\mathbf{r}_i$ w.r.t. the camera key pose vectors $\mathbf{c}_j$ and $\mathbf{c}_{j+1}$. It is straight-forward to find the analytic expressions for the sub-Jacobians $\mathbf{J}_{i,j}$, and $\mathbf{J}_{i,k}$ using basic differential calculus on $\mathbf{r}$, but some details of the derivation are worth mentioning:

- We use unit quaternions $\mathbf{q} = \left(\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\hat{\mathbf{n}}\right)$ to compute rotations, but parameterise the rotations using just the last three elements of $\mathbf{q}$.
- Each step of the processing chain is derived separately, and steps are concatenated using the chain rule.
- In the rolling shutter case, however, each individual residual has its own camera pose. This pose is a function of both the two nearest key poses, and of the observed image point.

4. Structure from Motion

The proposed bundle adjustment method is an essential part of the structure from motion (SfM) estimation pipeline shown in figure 2. In this section we provide details on how the other components of SfM are implemented in both the global shutter and the rolling shutter cases.

4.1. Point correspondences

All components in the SfM pipeline use inter-frame correspondences as measurements. These are found by first detecting interest points in each frame, using the FAST detector [26], and then tracking these with the KLT-tracker [20]. New points detected near existing trajectories are discarded in order to have the number of points fairly constant.

A first outlier rejection is done using *cross-checking* [5]. First points are tracked forward in time, and then the tracking is reversed. Only points that return to their original positions (within a threshold) are kept. This effectively removes most outliers from the tracker, without having to resort to global-shutter constraints such as homographies or fundamental/essential matrices. This is important, as these constraints are not satisfied under rolling-shutter geometry.

4.2. Global Shutter Structure from Motion

Here we describe the version of *global shutter structure from motion*, which we compare with rolling shutter SfM in the experiments. The description follows figure 2.

First, we build an initial geometry from three views with a sufficient relative baseline, using the five point method [23]. The essential matrices (and relative poses) between three views are robustly estimated in a RANSAC loop, and the 3D points are triangulated using the optimal method from [14]. The intermediate views are then added, and everything is bundled.

New views are successively added using the standard, sequential approach shown in figure 2. Here four views

are added at a time (this can be considered restrictive, as we are restricted to video input). First a PnP is applied, which minimises the L_2 error between the reprojected 3D model points, and the tracked points in the new frame that survived cross-checking (see section 4.1). This direct approach works well, as we can trust the 3D model points to be accurate. This is also exploited in PTAM [16].

Before new point tracks are added, a second level of outlier rejection is applied, using the scale-normalised standard deviation of multiple triangulations

$$\sigma_{\mathbf{X}} = \frac{1}{||\mathbf{t}_w - \boldsymbol{\mu}_{\mathbf{X}}||} \sqrt{\frac{1}{M-1} \sum_{m=1}^{M} (\mathbf{X}_m - \boldsymbol{\mu}_{\mathbf{X}})^2} \,. \quad (8)$$

The points $\mathbf{X}_m$ are triangulations between the first camera in a track, and all subsequent cameras where it is present, and $\boldsymbol{\mu}_{\mathbf{X}}$ is their mean. $\mathbf{t}_w$ is our frame of reference, chosen as the camera with the middle index in this sequence. Tracks where $\sigma_{\mathbf{X}}$ is above a threshold, are discarded.

Bundle adjustment (BA), as described in section 2 is applied, after the initial geometry has been estimated, as well as after all PnP and Triangulation steps. Running BA after PnP is especially important, as the outlier rejection step (8) relies on accurate poses.

4.3. Rolling Shutter Structure from Motion

Just like in the global shutter case, the rolling shutter aware SfM requires an initial estimate of structure and motion. For this we make use of the method suggested in [12], where the tracked points are first rectified using a 3D rotation model. This pre-rectification of points allows us to use global shutter geometric constraints, and consequently we then use the same initialisation as in the global shutter case, see section 4.2.

The original distorted points are however saved and subsequently used in the rolling shutter bundle adjustment, as described in section 3. The rolling shutter SfM follows the same scheme as the global shutter SfM (described in 4.2), but instead of using global shutter PnP, triangulation and bundle adjustment we use rolling shutter versions of these methods, as indicated in figure 2.

4.4. Rolling Shutter PnP and Triangulation

Ait-Aider *et al*. [2] solved the rolling shutter PnP problem by estimating the camera pose and linear camera motion during one frame. We instead propose to jointly estimate all the new poses. This allows us to exploit the coupling between poses, and thus constrain the problem better. The minimisation is done over the L_2 reprojection error between 3D points and tracked image points as with the global shutter PnP. Again, we use the RS camera model (7), with linear interpolation between camera positions, and SLERP interpolated rotations. This multi-frame PnP can be posed as the following optimisation problem:

$$\min_{\mathbf{c}_N,..,\mathbf{c}_{N+L}} \frac{1}{2} \sum_{j=N}^{N+L-1} \sum_{k \in \mathcal{V}_j} \text{dist}(\mathbf{p}_{j,k}, \mathbf{C}_j(y)\mathbf{X}_k)^2 \,. \quad (9)$$

Here N is index for the first of the new poses, L is the number of new views, and $\mathcal{V}_j$ is the index set of visible points in camera j, thus $\mathbf{X}_k, k \in \mathcal{V}_j$ is a 3D point, which is visible in camera j. Further, $\mathbf{p}_{j,k}$ is the observation of 3D point k in camera j and $\mathbf{C}_j(y) = \mathbf{C}(\mathbf{c}_j, \mathbf{c}_{j+1}, y)$ is an interpolated camera defined as in (7). Finally, $\text{dist}(\cdot,\cdot)$ is the Euclidean distance in image the plane.

We use the Levenberg-Marquardt algorithm, initialized with the previous camera, to solve (9). Note that an extra pose is estimated, $\mathbf{c}_{N+L}$. This pose does not have the same support as the rest of the parameters, and we currently discard it after estimation.

The rolling shutter aware triangulation method is similar to the classical optimal triangulation [14]. The only difference is that the two camera matrices are now a function of the current image row, see (7).

5. Experiments

In this section we describe our experimental setup for comparison of bundle adjustment methods on rolling shutter cameras. The evaluation is done by comparing the obtained camera trajectories to a reference trajectory.

5.1. Experiment Setup

All experiments use video from an iPhone 4 camera recorded at 1280×720 resolution, at 30 fps. Our evaluation is based on accurate reference trajectories, obtained using a second camera that has a global shutter, a Canon S95 with 1280×720 resolution, at 24 fps. This prosumer compact camera produces good image quality due to a relatively large image sensor (1/1.7"). This combined with the wide angle lens allows very accurate camera trajectory estimation. The reprojection error is around 0.2 pixel which is around a third of the error for the rolling shutter SfM on the iPhone data. For the iPhone 4 readout time, we use the value of 32.37 msec, as listed in [25].

The two cameras are rigidly mounted on a rig, with overlapping fields of view, and optical centers as close as possible, see figure 4. Both cameras are calibrated for intrinsic camera parameters, radial, and tangential distorsions, allowing us to use calibrated epipolar geometry throughout.

For frame-accurate synchronization, we start and end each recording with a snap of fingers, which is visible in both cameras. The maximal error in this synchronization is a function of the lower of the two frame rates. Here we get $1/24$ sec as maximal error, which is small compared to the length of our sequences (typically around 4 seconds).

Figure 4. Left: Camera rig used in experiments. Right: Synchronization procedure.

5.2. Trajectory Comparison

In order to compare an estimated trajectory with the ground truth, an alignment is needed. First we need to temporally align the trajectories, and then estimate the unknown rotation, translation and scale between the two trajectories. The synchronisation procedure gives us time-stamps on all trajectory points, and we use these to re-sample (linearly) each test-trajectory to temporally align it with the ground truth trajectory $\{\mathbf{G}_j\}_1^N$. After this, the resampled trajectory $\{\mathbf{X}\}_1^N$ is spatially aligned to the ground-truth, by minimising the sum of all squared point-to-point distances:

$$\min_{s,\mathbf{R},\mathbf{t}} \sum_{k=1}^{N} ||\mathbf{G}_k - s\mathbf{R}(\mathbf{X}_k - \mathbf{t})||^2 . \tag{10}$$

The geometric error of a camera pose estimate consists of two components: a translation error and a rotation error. The two errors are difficult to combine in a generic way, and we have chosen to focus on an analysis of the translation error. The translation error measure used here is based on the area of the surface between the curves, as suggested in [15], but we use a discretized version:

$$\varepsilon = \sum_{k=1}^{N-1} \text{tri}(\mathbf{G}_k, \tilde{\mathbf{X}}_k, \tilde{\mathbf{X}}_{k+1}) + \text{tri}(\mathbf{G}_k, \mathbf{G}_{k+1}, \tilde{\mathbf{X}}_{k+1}) . \tag{11}$$

Here $\tilde{\mathbf{X}}_k = s\mathbf{R}(\mathbf{X}_k - \mathbf{t})$, and the function $\text{tri}(\cdot,\cdot,\cdot)$ computes the area of the triangle defined by its three arguments.

5.3. Evaluation Sequences

We have collected a set of 36 sequences using the rig in figure 4. Frames from a subset of the sequences are shown in figure 5. As can be seen, the sequences have great variability in scene content. The camera motions in the sequences are various mixtures of three fundamental motion types: **FORWARD**, **SIDEWAYS**, and **3D ROTATION**.

5.4. Compared Methods

We compare the following methods:

- **GSBA**: Global Shutter Bundle Adjustment.

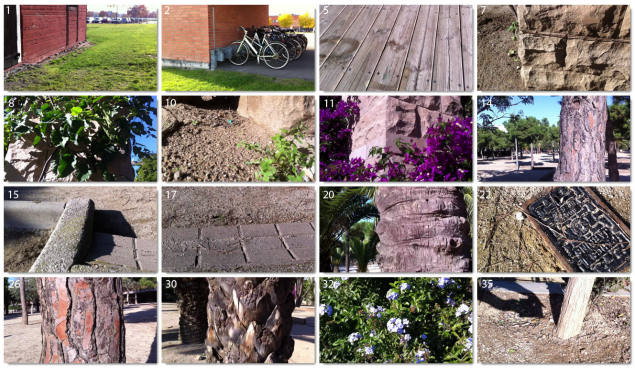


Figure 5. Sample images from 16 of the 36 test sequences.

- **PRBA**: BA on pre-rectified point tracks, using a 3D rotational model [8]. This corresponds roughly to the approach suggested in [12].
- **PRBA-T**: With triangulation outlier rejection applied, according to (8).
- **RSBA**: The Rolling Shutter Bundle Adjustment method proposed in this paper.

6. Results

The results obtained with the four different methods are documented in three different ways. We illustrate the geometric quality of the respective results by showing several plots of camera trajectories. For a quantitative comparison, we compute numeric errors for the respective approaches. Finally, we compare the execution speeds of our **RSBA** to that of **GSBA**.

6.1. Camera Trajectory Accuracy

The accuracy is evaluated using the spanned area between the ground truth and the evaluated trajectory, as defined in (11). This translation error can easily be illustrated using 2D projections of the estimated 3D camera trajectories that are estimated with the respective method.

We have plotted the results for four of the sequences in figure 6. More plots are in the supplementary material.

The numerical results are collected in table 1, the lowest error for each sequence is shown in boldface. We have also plotted the relative improvement of the other methods compared to **GSBA** (cf. figure 7) and to **PRBA** (cf. figure 8). The relative scores are sorted and the respective trajectory characteristics are given below the respective plot. As can be seen, there is no systematic correlation between the type of motion and the amount of improvement of the result for **RSBA**. We thus conclude that the improvement from using **RSBA** is not confined to any of these motion categories, but instead applies to all of them.

Compared to **GSBA**, the **PRBA** method from [12] often produces worse results, see figure 7. We suspected that

seq#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
RSBA	0.042	0.124	**0.096**	0.129	**0.088**	0.126	**0.019**	**0.500**	**0.154**	**0.534**	**0.032**	**0.091**	**0.116**	**0.029**	**0.066**	**0.179**	**0.055**	**0.591**
PRBA-T	**0.039**	**0.101**	0.125	**0.128**	0.968	0.247	0.033	2.600	0.154	1.110	0.068	0.192	0.240	0.286	0.092	0.208	0.060	3.280
GSBA	0.341	0.230	0.154	0.230	0.680	**0.122**	0.085	1.960	0.271	1.080	0.082	0.217	0.471	0.203	0.154	0.308	0.135	0.679
PRBA	0.182	1.150	0.127	2.860	0.988	0.247	0.033	2.590	0.154	1.360	0.066	0.194	0.242	0.344	0.093	0.208	0.060	3.280

seq#	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
RSBA	4.590	**0.049**	**0.018**	**0.123**	**0.089**	**0.244**	**0.387**	**0.425**	**0.061**	**0.584**	**0.102**	**0.309**	0.701	**0.060**	**0.191**	**0.068**	**0.036**	**0.026**
PRBA-T	**2.370**	0.322	0.061	3.930	0.131	0.407	0.502	1.520	0.238	3.320	0.104	6.990	1.360	0.233	1.080	0.153	0.083	0.039
GSBA	135.0	0.235	0.123	0.374	0.170	0.785	0.886	0.585	0.129	0.952	0.235	0.714	**0.679**	0.180	0.798	0.204	0.110	0.071
PRBA	2.370	0.328	0.078	1.140	0.185	0.476	0.539	1.340	0.269	3.330	0.644	7.390	16.00	0.233	1.090	0.153	0.081	0.186

Table 1. Error of the trajectories estimated from rolling shutter sequences against estimates from global shutter sequences.

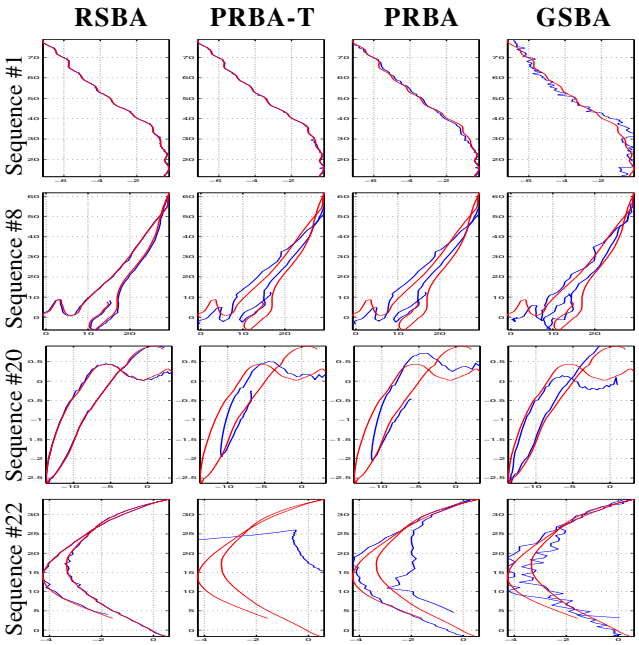


Figure 6. Examples of 2D projections of the 3D camera trajectories for the different methods. RED: ground-truth, BLUE: test results.

the cause of this were outliers in the 3D point cloud, and thus included the **PRBA-T** method in the evaluation. As can be seen in figure 8, the outlier rejection improves results in many cases, but has only been partly successful. Note that in [12] **PRBA** was reported to be consistently better than **GSBA**. A likely explanation for the different results is the difference in datasets; our new dataset shows a much broader variation in terms of motion and scene complexity than the one in [12]. There are also differences in the implementation used, where in [12] the SBA solver is used, and here we have implemented our own bundler. The main implementation difference is the parametrization: we use six parameters in the camera model and SBA uses seven. We also noted that by applying a preconditioner the iteration count could be reduced (see section 2.1).

The proposed method **RSBA**, however, performs significantly better in most of the cases: Accuracy is improved by more than 50% in about 75% of the sequences. Only in very few cases, no improvement of accuracy has been observed.

A caveat is due here: The sequences collected here have hand-held camera motions. Sequences collected with stabilisation rigs are quite different in nature (the camera motion is much smoother), and whether **RSBA** improves the accuracy in such cases is currently unknown.

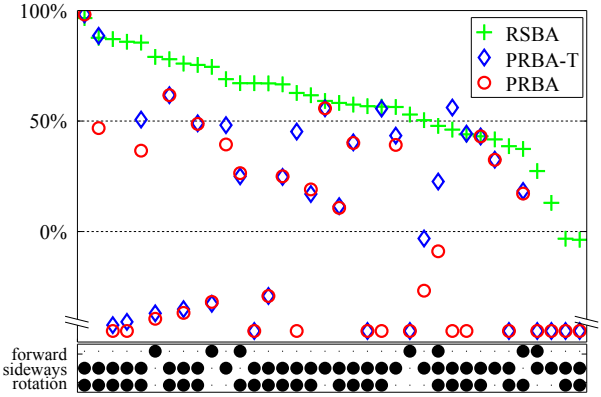


Figure 7. Improvement (in percentage) of methods **RSBA**, **PRBA-T**, and **PRBA** compared to **GSBA**.

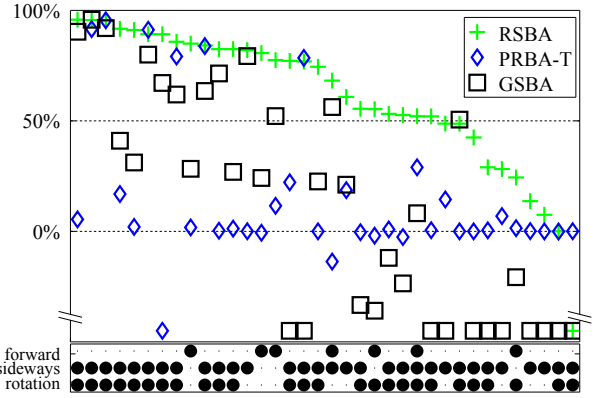


Figure 8. Improvement (in percentage) of methods **RSBA**, **PRBA-T**, and **GSBA** compared to **PRBA**.

6.2. Computational Speed

Our **GSBA** run on a sequence with 221 cameras computed 13 777 structure points in 5 iterations. This took on average 4.91 sec on a W3520 Intel PC. Our **RSBA** run on the same sequence computed 14 320 structure points in 4 iterations, and 8.37 sec, on average. On a smaller system with 74 cameras, **RSBA** took 2.36 sec for 5 602 points, while **GSBA** took 1.49 sec for 5 577 points. We have also made preliminary comparisons with **SBA** [19]. On many sequences our **GSBA** solver has similar complexity, but on difficult sequences (e.g. with rolling shutter), our use of a preconditioner leads to faster convergence.

7. Conclusions

In this paper, we have presented the **RSBA**, to the best of our knowledge the first bundle adjustment system that explicitly models rolling shutter geometry. Compared to global shutter BA, the increase in computation time is moderate. Using real image sequences captured with an iPhone 4, we have demonstrated that our proposed method consistently improves the accuracy of SfM across a wide variety of camera motions. This is a first attempt at rolling shutter bundle adjustment, and there are many things that can be improved. In the future we plan to investigate other trajectory representations, and other cost functions that are not based on the L_2 norm.

Acknowledgements This work has been supported by ELLIIT, the Strategic Area for ICT research, funded by the Swedish Government, and from VPS, funded by the Swedish Foundation for Strategic Research. The CENIIT organisation at LiTH, the Swedish Research Council through a grant for the project *Embodied Visual Object Recognition*, and by Linköping University.

References

[1] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski. Bundle adjustment in the large. In *ECCV'10*. 1, 2, 3

[2] O. Ait-Aider, N. Andreff, J. M. Lavest, and P. Martinet. Simultaneous object pose and velocity computation using a single view from a rolling shutter camera. In *ECCV'06*, May 2006. 2, 3, 5

[3] O. Ait-Aider, A. Bartoli, and N. Andreff. Kinematics from lines in a single rolling shutter image. In *CVPR'07*, Minneapolis, USA, June 2007. 2

[4] O. Ait-Aider and F. Berry. Structure and kinematics triangulation with a rolling shutter stereo rig. In *ICCV*, 2009. 2

[5] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. In *IEEE ICCV*, Rio de Janeiro, Brazil, 2007. 4

[6] C. Engels, H. Stewénius, and D. Nistér. Bundle adjustment rules. In *Photogrammetric Computer Vision*, 2006. 1, 2, 3

[7] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24:381–395, June 1981. 2

[8] P.-E. Forssén and E. Ringaby. Rectifying rolling shutter video from hand-held devices. In *CVPR'10*. 3, 6

[9] J.-M. Frahm, P. Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys. Building rome on a cloudless day. In *ECCV'10*, 2010. 1

[10] C. Geyer, M. Meingast, and S. Sastry. Geometric models of rolling-shutter cameras. In *6th OmniVis WS*, 2005. 1

[11] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. 1

[12] J. Hedborg, E. Ringaby, P.-E. Forssén, and M. Felsberg. Structure and motion estimation from rolling shutter video. In *IWMV workshop at ICCV'11*, 2011. 1, 2, 5, 6, 7

[13] Y. Jeong, D. Nistér, D. Steedly, R. Szeliski, and I.-S. Kweon. Pushing the envelope of modern methods for bundle adjustment. In *CVPR'10*, June 2010. 1, 2, 4

[14] K. Kanatani, Y. Sugaya, and H. Niitsuma. Triangulation from two views revisited: Hartley-sturm vs. optimal correction. In *BMVC*, pages 173–182, 2008. 4, 5

[15] K. Kishimoto. On a distance between two curves. In *First Int. Symp. for Science on Form*, pages 121–128, 1986. 6

[16] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *ISMAR'09*, October 2009. 1, 2, 3, 5

[17] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly Journal of Applied Mathmatics*, II(2):164–168, 1944. 2

[18] F. Liu, M. Gleicher, J. Wang, H. Jin, and A. Agarwala. Subspace video stabilization. *ACM ToG*, 30(1), 2011. 1

[19] M. A. Lourakis and A. Argyros. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Trans. Math. Software*, 36(1):1–30, 2009. 1, 2, 8

[20] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI'81*, pages 674–679, 1981. 4

[21] K. Madsen, H. B. Nielsen, and O. Tingleff. Methods for non-linear least squares problems, 2nd ed. Technical report, Technical University of Denmark, April 2004. 2

[22] D. Martinec and T. Pajdla. Robust rotation and translation estimation in multiview reconstruction. In *CVPR'07*. 1

[23] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE TPAMI*, 6(26):756–770, June 2004. 4

[24] M. Pollefeys, L. van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *IJCV*, 59(3):207–232, 2004. 1

[25] E. Ringaby and P.-E. Forssén. Efficient video rectification and stabilisation for cell-phones. *IJCV*, Online June 2011. 5

[26] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *ECCV'06*, May 2006. 4

[27] K. Shoemake. Animating rotation with quaternion curves. In *Int. Conf. on CGIT*, pages 245–254, 1985. 3

[28] G. Thalin. Camera rolling shutter amounts. `http://www.guthspot.se/video/deshaker.htm`. 1

[29] B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. In *Vision Algorithms: Theory and Practice*, pages 298–375, 2000. 1, 2