Rolling-Shutter-Aware Differential SfM and Image Rectification

Manuel Fritsche
ETH Zürich
manuelf@ethz.ch

Felix Graule
ETH Zürich
graulef@ethz.ch

Thomas Ziegler
ETH Zürich
zieglert@ethz.ch

Abstract

In this work we implement a method to rectify the distortion of a Rolling-Shutter camera and estimate its relative motion. A modified differential Structure-from-Motion (SfM) algorithm proposed by Zhuang et al. [24] is used to correct this distortion. We provide a detailed description and present experimental results on both synthetic and real world data. We show that the approach results in good rectification given an accurate optical flow estimate.

1. Introduction

Many consumer-grade cameras (e.g. smartphone cameras) are Rolling-Shutter (RS) cameras. In contrast to Global-Shutter (GS) cameras, which read out all pixels jointly, RS cameras capture the image line-by-line. This reduces manufacturing cost significantly. However, computer vision algorithms like epipolar geometry [10] and SfM [7] assume a GS camera model. Therefore, their accuracy might be reduced when using RS cameras due to the so-called RS distortion: for a moving RS camera each scanline has its own optical center. Thus, the image is distorted.

In this work we implement a method proposed by Zhuang *et al*. [24] to rectify the RS distortion and jointly estimate the camera's motion. Using a suitable motion model the need for additional parameters is eliminated. Assuming a camera movement with constant velocity, the RS distortion is rectified using a linear model applied on the optical flow. Assuming a constant acceleration movement of the camera, a 9-point algorithm is used to solve for the correction factors on the optical flow. In a nonlinear refinement step the initial estimates are optimized. Then, we can backproject the RS distorted frame to get the initial GS frame. The overall idea of the method is depicted in figure 1.

The goal of this work is to verify the proposed method and to provide a detailed description on how to implement it. We made our implementation available on GitHub[1].

[1]https://github.com/ThomasZiegler/RS-aware-differential-SfM

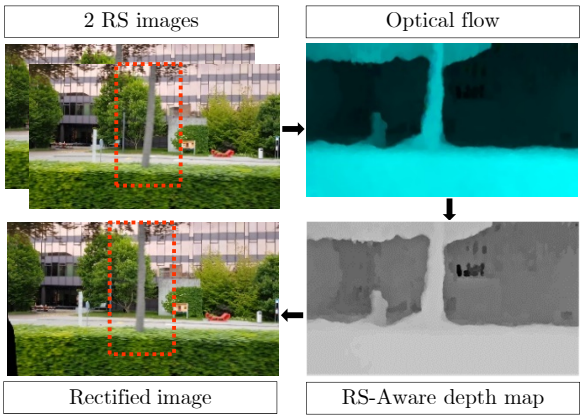

Figure 1. Overall procedure: Rectify RS distortion and estimate camera motion based on optical flow between two RS frames. Due to the RS distortion the tree trunk is slanted.

2. Related Work

The RS distortion problem in single and multi-view images is part of numerous studies. Saurer *et al*. [22] show that large-scale SfM with RS images is possible.

However, it is challenging to estimate the camera trajectory in RS SfM. Dai *et al*. [6] proposed a method using a high number of image correspondences. This reduces the practicability when using RANSAC. Other methods bypass the initial relative pose estimation or simplify it. Hanning *et al*. [9] and Karpenko *et al*. [13] correct the RS effect of a smartphone video by using the integrated IMU to get a camera trajectory estimate. Purkait and Zach [18] restrict the possible camera trajectory by using a specific car motion model. Saurer *et al*. [21] optimize for the absolute pose under the simplifying assumption of zero angular velocity. Albl *et al*. [2] use a double linearized RS camera model which requires a good initial estimate for the camera orientation. Purkait *et al*. [19] and Lao *et al*. [14] leverage geometric properties of the 3D scene in the images, in particular vanishing directions and straightness constraints. Under the assumption of a uniform camera velocity, Meilland *et al*. [16] proposed a SfM algorithm jointly solving for the RS distortion and motion blur using RGBD sensor data.

3. Method

3.1. Notation and motion model

We define $\mathbf{X} \equiv [X_x, X_y, X_z]^\top$ and $\mathbf{x} \equiv [x_x, x_y, 1]^\top$ as world and image coordinates, respectively. The transformation between them is given by

$$\lambda \mathbf{x} = \mathbf{R}\mathbf{X} + \mathbf{t} = \left[\mathbf{R} \mid \mathbf{t}\right] \left[\mathbf{X}^\top \mid 1\right]^\top \tag{1}$$

with $\lambda \in \mathbb{R}\backslash\{0\}$ and $\mathbf{R}$, $\mathbf{t}$ being a rotation matrix and translation vector, respectively.

As described by Murray *et al*. [17] the rigid motion of the camera follows

$$\frac{d}{dt}\mathbf{x} = \boldsymbol{\omega} \times \mathbf{X} + \mathbf{v}, \tag{2}$$

where $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^\top \in \mathbb{R}^3$ is the angular velocity and $\mathbf{v} = [v_x, v_y, v_z]^\top \in \mathbb{R}^3$ the linear velocity of the camera. The angular velocity can be mapped into a rotation matrix via the the exponential map $\mathbf{exp}(.) : \mathfrak{so}(3) \rightarrow SO(3)$:

$$\mathbf{R} = \mathbf{exp}(\hat{\boldsymbol{\omega}}). \tag{3}$$

Here, $SO(3)$ is the so-called *special orthogonal group* and $\mathfrak{so}(3)$ the corresponding lie algebra [4]. It is also referred to as the space of 3×3 skew symmetric matrices [17]. For a vector $\mathbf{p} = [p_1, p_2, p_3]^\top \in \mathbb{R}^3$, we define $\hat{\mathbf{p}} \in \mathfrak{so}(3)$ as:

$$\hat{\mathbf{p}} \equiv \begin{bmatrix} 0 & -p_3 & p_2 \\ p_3 & 0 & -p_1 \\ -p_2 & p_1 & 0 \end{bmatrix}. \tag{4}$$

3.2. Differential SfM

As described by Jepson and Heeger [12], using the linear velocity $\mathbf{v}$ and angular velocity $\boldsymbol{\omega}$ the optical flow $\mathbf{u}^{(i)}$ of a pixel $\mathbf{x}^{(i)} = [x_x^{(i)}, x_y^{(i)}, 1]^\top$ can be expressed as

$$\mathbf{u}^{(i)} = \frac{\mathbf{A}^{(i)}\mathbf{v}}{X_z^{(i)}} + \mathbf{B}^{(i)}\,\boldsymbol{\omega} \tag{5}$$

with

$$\mathbf{A}^{(i)} = \begin{bmatrix} 1 & 0 & -x_x^{(i)} \\ 0 & 1 & -x_y^{(i)} \end{bmatrix} \tag{6}$$

and

$$\mathbf{B}^{(i)} = \begin{bmatrix} -x_x^{(i)}x_y^{(i)} & (1 + x_x^{(i)}x_x^{(i)}) & -x_y^{(i)} \\ -(1 + x_y^{(i)}x_y^{(i)}) & x_x^{(i)}x_y^{(i)} & x_x^{(i)} \end{bmatrix}. \tag{7}$$

To simplify the notation, in the following we drop the $^{(i)}$ index, if a formula only depends on a single pixel i.

Following the work of Ma *et al*. [15] we can use (5) to rewrite the differential epipolar constraint as

$$\mathbf{u}^\top \hat{\mathbf{v}}\mathbf{x} + \mathbf{x}^\top \mathbf{S}\mathbf{x} = 0 \tag{8}$$

with the symmetric matrix $\mathbf{S} = \frac{1}{2}\left(\hat{\mathbf{v}}\hat{\boldsymbol{\omega}} + \hat{\boldsymbol{\omega}}\hat{\mathbf{v}}\right)$. This equation yields a constraint for the image points which can be used to solve for $\mathbf{v}$ and $\boldsymbol{\omega}$.

3.3. RS-Aware-Differential SfM

Using a RS camera, each scanline of every frame has its own pose. We assume the RS camera jointly scans the pixels on the x-axis. Hence, each y-coordinate corresponds to a scanline as illustrated in figure 2.

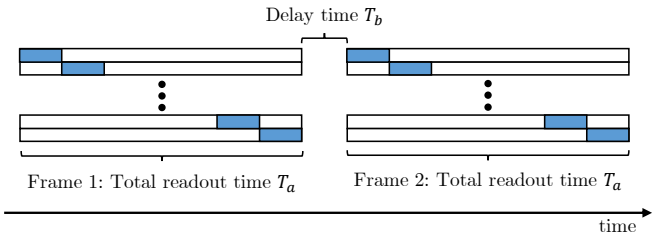

Figure 2. Two consecutive frames taken with a RS camera. The lines correspond to the scanlines, with the readout time marked in blue.

Given two images and a 3D point mapped to image coordinates $\mathbf{x}^{(i)}$ in image 1 and $\mathbf{x}^{(j)}$ in image 2, the optical flow is $\mathbf{u}^{(i)} = [x_x^{(j)} - x_x^{(i)}, x_y^{(j)} - x_y^{(i)}]^\top$. As shown by Zhuang *et al*. [24], the relative translation $\mathbf{t}_{ij}$ and rotation $\mathbf{r}_{ij}$ between the two scanlines $x_y^{(i)}$ and $x_y^{(j)}$ can be written as

$$\mathbf{t}_{ij} = \beta^{(i)}\Delta\mathbf{t}, \qquad \mathbf{r}_{ij} = \beta^{(i)}\Delta\mathbf{r}. \tag{9}$$

Here, $\Delta\mathbf{t}$ and $\Delta\mathbf{r}$ are the relative translation and rotation of the camera between the first scanline of image 1 and image 2. The factor $\beta^{(i)}$ depends on the displacement of the scanlines in the optical flow, the readout time ratio $\gamma = \frac{T_a}{T_a+T_b}$ and an acceleration factor k.

3.3.1 Constant Velocity

Throughout this project two assumptions for the camera movement are made. First, a camera motion with constant velocity between two frames is assumed. The factor $\beta^{(i)}$ is then equal to

$$\alpha^{(i)} = 1 + \frac{\gamma}{h}(x_y^{(j)} - x_y^{(i)}), \tag{10}$$

where h relates to the number of scanlines in an image.

3.3.2 Constant Acceleration

For the second case, a camera motion with constant acceleration between two frames is assumed. This introduces an additional parameter, namely the acceleration factor k. The factor $\beta^{(i)}$ can then be found as:

$$\beta^{(i)}(k) = \left(\alpha^{(i)} + k\tilde{\alpha}^{(i)}\right)\frac{2}{2+k} \tag{11}$$

with

$$\tilde{\alpha}^{(i)} = \frac{1}{2}\left(\left(1 + \frac{\gamma x_y^{(j)}}{h}\right)^2 - \left(\frac{\gamma x_y^{(i)}}{h}\right)^2\right). \tag{12}$$

For $k=0$ the constant acceleration case reduces to the constant velocity case. Thus, the algorithms are described for the constant acceleration case in the following.

3.3.3 RS Differential Epipolar Constraint

By assuming a small motion between two images, the velocities $\mathbf{v}$ and $\boldsymbol{\omega}$ can be used as an approximation for the displacement $\Delta\mathbf{t}$ and $\Delta\mathbf{r}$. Thus, the factor β can be used to correct (5), leading to

$$\mathbf{u} = \beta \left(\frac{\mathbf{A}\mathbf{v}}{X_z} + \mathbf{B}\,\boldsymbol{\omega} \right). \tag{13}$$

Here, $\mathbf{v}$ and $\boldsymbol{\omega}$ describe the relative pose between the first scanlines of the two RS frames. Furthermore, the differential epipolar constraint (8) can be corrected for the RS distortion, resulting in

$$\mathbf{u}^\top \hat{\mathbf{v}}\mathbf{x} + \beta \mathbf{x}^\top \mathbf{S}\mathbf{x} = 0. \tag{14}$$

3.4. Rectification

Let $(\mathbf{t}_{1i}, \mathbf{r}_{1i})$ be the relative poses between the first and the i-th scanline within the same image. With $\boldsymbol{\omega}$, $\mathbf{v}$ and the correct depth values, it is possible to retrieve the relative camera poses for each scanline as shown by Zhuang *et al*. [24]. Using the first scanline of the first RS image as the origin, the relative poses can be derived as:

$$\mathbf{t}_{1i} = \left(\frac{\gamma\, x_y^{(i)}}{h} + \frac{k}{2} \left(\frac{\gamma\, x_y^{(i)}}{h} \right)^2 \right) \mathbf{v}, \tag{15}$$

$$\mathbf{r}_{1i} = \left(\frac{\gamma\, x_y^{(i)}}{h} + \frac{k}{2} \left(\frac{\gamma\, x_y^{(i)}}{h} \right)^2 \right) \boldsymbol{\omega}. \tag{16}$$

The pose of the i-th scanline is $\mathbf{t}_i = \mathbf{t}_{1i}$ and $\mathbf{R}_i = \mathbf{exp}(\hat{\mathbf{r}}_{1i})$. With the known poses and (1) it is possible to backproject each pixel of the first RS frame into the 3D world. Using the same equation one can backproject each 3D point into the image plane assuming the pose corresponding to the first scanline. Since only the first scanline pose is used, this result approximates a GS image.

4. Implementation

The pipeline is implemented using C++, Eigen [8] and OpenCV [5]. The work was distributed as follows:

Manuel Fritsche implemented RANSAC and the Minimal Solver, including the implementation of the approach presented by Ma *et al*. [15] and the derivations in section 4.3.

Felix Graule generated both synthetic and real world data. Furthermore, he took care of the Optical Flow calculation for real world data using DeepFlow [23].

Thomas Ziegler implemented the Depth Estimation, the Nonlinear Refinement and the Image Rectifier. He also worked out how to calculate the Optical Flow on synthetic data.

4.1. Pipeline

The algorithm consists of several steps executed sequentially as shown in figure 3. In the following sections each of the steps is described in more detail.

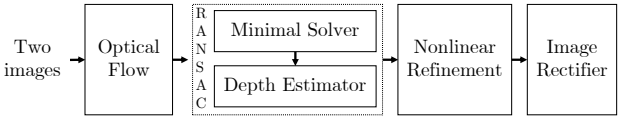

Figure 3. Each box marks a step in the pipeline. The Minimal Solver and the Depth Estimator are executed multiple times within RANSAC.

4.2. Optical Flow

The input consists of two consecutive images that were taken by a rolling-shutter camera. Retrieving the optical flow is done by two separate methods. In the case of synthetic images (section 5.1) we use the information provided by the renderer to calculate the exact ground truth flow. For real world images (section 5.2) this is not possible. Instead, the pipeline uses the optical flow implementation DeepFlow [23] as described in section 4.2.2.

4.2.1 Ground Truth Flow

For each pixel the renderer provides the corresponding 3D point in the world frame. One can project the 3D points from the first RS image into the second RS image using (1). However, as described in section 3.3 each scanline has its own pose. Hence, one does not know beforehand in which x_y coordinate the point is projected. Thus, we do not know which scanline pose to use upfront. We therefore calculate the corresponding image coordinates using all scanline poses. From all possible solutions we choose the one with the smallest deviation between x_y and the corresponding scanline.

4.2.2 DeepFlow

DeepFlow [23] allows to robustly estimate the optical flow between two frames even for large displacements. It uses a multi-layer neural network tailored to recognize optical flow. DeepFlow is implemented in OpenCV's extra modules in the class *optflow*. The implementation is fully preconfigured and thus easy-to-use as no parameter-tuning is needed.

4.3. Minimal Solver

The Minimal Solver gets the optical flow as its input and uses the differential epipolar constraint in (14) to estimate

$\mathbf{v}, \boldsymbol{\omega}$ and k. There are 9 unknowns in this equation:

$$\mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}, \quad \mathbf{S} = \frac{1}{2}\left(\hat{\mathbf{v}}\hat{\boldsymbol{\omega}} + \hat{\boldsymbol{\omega}}\hat{\mathbf{v}}\right) = \begin{bmatrix} s_1 & s_2 & s_3 \\ s_2 & s_4 & s_5 \\ s_3 & s_5 & s_6 \end{bmatrix}. \tag{17}$$

Using 9 image relations from the optical flow, a 9×9 matrix $\mathbf{Z}(k)$ is built from the epipolar constraint in (14), where

$$\mathbf{Z}(k)\mathbf{e} = 0. \tag{18}$$

The 9×1 vector $\mathbf{e}$ contains all unknowns in the constraint:

$$\mathbf{e} = [v_x, v_y, v_z, s_1, s_2, s_3, s_4, s_5, s_6]^\top. \tag{19}$$

By using (14), $\mathbf{Z}(k)$ can be found as:

$$\mathbf{Z}(k) = [\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_9]^\top \tag{20}$$

Here, $\mathbf{z}_i$ depends on the optical flow $\mathbf{u}^{(i)} = [u_x^{(i)}, u_y^{(i)}]^\top$, the image coordinates $\mathbf{x}^{(i)} = [x_x^{(i)}, x_y^{(i)}]^\top$ and $\beta^{(i)}(k)$:

$$\mathbf{z}_i = \begin{bmatrix} x_y^{(i)} - u_y^{(i)} \\ -x_x^{(i)} + u_x^{(i)} \\ x_x^{(i)}u_y^{(i)} - x_y^{(i)}u_x^{(i)} \\ -\beta^{(i)}(k)\left(x_x^{(i)}\right)^2 \\ -2\beta^{(i)}(k)x_x^{(i)}x_y^{(i)} \\ -2\beta^{(i)}(k)x_x^{(i)} \\ -\beta^{(i)}(k)\left(x_y^{(i)}\right)^2 \\ -2\beta^{(i)}(k)x_y^{(i)} \\ -\beta^{(i)}(k) \end{bmatrix}. \tag{21}$$

As mentioned before, k is equal to 0 in the constant velocity case. Thus, $\beta^{(i)}(0) = \alpha^{(i)}$ is just a constant which can be absorbed in the optical flow $\mathbf{u}$. However, in the constant acceleration case the acceleration factor k has to be estimated. Using the fact that (18) only has non-trivial solutions if $\mathbf{Z}(k)$ is rank deficient, k can be estimated from

$$\det(\mathbf{Z}(k)) = 0. \tag{22}$$

This yields a polynomial, which Zhuang *et al*. [24] suggest to solve directly. However, this requires to solve (22) symbolically, which is non-trivial. Thus, we propose an alternative solution using eigenvalues in the following. The matrix $\mathbf{Z}(k)$ can be decomposed into 3×3 block matrices of size 3×3:

$$\mathbf{Z}(k) = \begin{bmatrix} \mathbf{A} & \mathbf{B}(k) & \mathbf{C}(k) \\ \mathbf{D} & \mathbf{E}(k) & \mathbf{F}(k) \\ \mathbf{G} & \mathbf{H}(k) & \mathbf{J}(k) \end{bmatrix}. \tag{23}$$

We can then rewrite (22) as:

$$\det(\mathbf{Z}(k)) = \det(\mathbf{A})\det\left(\mathbf{M}(k)\right) = 0 \tag{24}$$

with

$$\mathbf{M}(k) = \begin{bmatrix} \mathbf{E}(k) - \mathbf{D}\mathbf{A}^{-1}\mathbf{B}(k) & \mathbf{F}(k) - \mathbf{D}\mathbf{A}^{-1}\mathbf{C}(k) \\ \mathbf{H}(k) - \mathbf{G}\mathbf{A}^{-1}\mathbf{B}(k) & \mathbf{J}(k) - \mathbf{G}\mathbf{A}^{-1}\mathbf{C}(k) \end{bmatrix}. \tag{25}$$

Now every entry in $\mathbf{M}(k)$ depends on a linear combination of $\beta^{(i)}(k) = (\alpha^{(i)} + k\tilde{\alpha}^{(i)})\frac{k}{2+k}$ for different i. This allows the following decomposition of $\mathbf{M}(k)$:

$$\mathbf{M}(k) = (\mathbf{P} + k\mathbf{Q})\frac{k}{2+k}, \tag{26}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are not dependent on k.

Now for $k \neq 0$, $k \neq -2$ and an invertible matrix $\mathbf{Q}$, we can rewrite (22) as:

$$\det(\mathbf{P} + k\mathbf{Q}) = \det(-\mathbf{Q})\det(-\mathbf{Q}^{-1}\mathbf{P} - k\mathbf{I}) = 0. \tag{27}$$

Thus, the k values that solve (27) are the eigenvalues of $-\mathbf{Q}^{-1}\mathbf{P}$. From the multiple k values returned, the one with the smallest absolute value is used. We pick this value to favor small accelerations.

After k is estimated, $\beta^{(i)}(k)$ is treated as a constant factor, equivalent to the constant velocity case. It can then be absorbed in the optical flow, which allows us to estimate the velocities $\mathbf{v}$ and $\boldsymbol{\omega}$ with the 9-point algorithm described by Ma *et al*. [15].

4.4. Depth Estimation

The Minimal Solver provides an initial estimate of $\mathbf{v}$ and $\boldsymbol{\omega}$. Depth Estimation is done by solving for the optimal depth value $X_z^{(i)}$ for each pixel i. Using (13) we get the following optimization

$$\underset{\boldsymbol{Z}}{\arg\min}\left\{\sum_{i\in\boldsymbol{O}}\left\|\mathbf{u}^{(i)} - \beta^{(i)}(k)\left(\frac{\mathbf{A}^{(i)}\mathbf{v}}{X_z^{(i)}} + \mathbf{B}^{(i)}\,\boldsymbol{\omega}\right)\right\|_2^2\right\}, \tag{28}$$

where $\boldsymbol{Z} = \{X_z^{(1)}, X_z^{(2)}, \cdots\}$ is the set of all depth values and $\boldsymbol{O}$ the set of all pixels. The optimization is independent for each pixel and can be done over the whole set at once resulting in an algebraic minimum. We use Google Ceres solver [1] to perform this optimization.

4.5. RANSAC

RANSAC is used to make the algorithm robust against outliers. During each trial 9 image points are sampled without replacement. On these the Minimal Solver is executed to estimate $\mathbf{v}$, $\boldsymbol{\omega}$ and k. Subsequently, the Depth Estimator calculates the depth values $X_z^{(i)}$ of all image points, which are then used to determine the inliers. The estimation error of each point i is defined using the optical flow and (13):

$$\left\|\mathbf{u}^{(i)} - \beta^{(i)}(k)\left(\frac{\mathbf{A}^{(i)}\mathbf{v}}{X_z^{(i)}} + \mathbf{B}^{(i)}\,\boldsymbol{\omega}\right)\right\|_2^2. \tag{29}$$

4.6. Nonlinear Refinement

The result from RANSAC described in 4.5 provides an algebraic minimum of (13). However, for a more accurate solution the geometric reprojection error should be minimized. This is done using

$$\underset{k,\mathbf{v},\boldsymbol{\omega},\boldsymbol{Z}}{\arg\min}\left\{\sum_{i\in\boldsymbol{O}}\left\|\mathbf{u}^{(i)}-\beta^{(i)}(k)\Big(\frac{\mathbf{A}^{(i)}\mathbf{v}}{X_z^{(i)}}+\mathbf{B}^{(i)}\,\boldsymbol{\omega}\Big)\right\|_2^2\right\}, \tag{30}$$

where $\boldsymbol{O}$ is the set of RANSAC inliers and $\boldsymbol{Z}$ the set of corresponding depth values for a pixel in the set $\boldsymbol{O}$. In contrast to (28) we not only optimize over the depth values but also over all motion parameters. Note that $\mathbf{v}$ can only be determined up to scale. Thus, $X_z^{(i)}$ can only be estimated up to scale. We again use Google Ceres solver [1] for this optimization.

5. Experiments and Results

5.1. Synthetic Data Generation

We generated synthetic data using a 3D renderer for Matlab [11] and the textured 3D model *castle* [20]. First, the trajectory of the virtual camera is calculated using (9) leading to a tuple $(\mathbf{R}_i, \mathbf{t}_i)$ for every scanline. We then iteratively render a GS image for every scanline pose, while only storing the horizontal line of the rendering corresponding to the current scanline. Finally, all lines are fused together to form one RS image. After moving the virtual camera to the next initial position this procedure is repeated to get the second RS image. Through this method we can fully control all motion parameters, minimize the error induced by imprecise optical flow and compare the results to ground truth.

5.2. Real World Data Generation

Using real world data we show how our implementation performs in practically relevant settings. The images were captured using the rear camera of a Samsung Galaxy S8. The camera was calibrated using the *Single Camera Calibrator App* in Matlab. Auto-focus was turned off to ensure constant camera intrinsics in the videos. To achieve a significant RS distortion a fast motion is required. Hence, we recorded the test sequences from ETH's shuttle buses.

5.3. Performance on Synthetic Data

The effectiveness of our pipeline can be seen in figure 4. Looking at the edges marked by red lines, we see how the previously slanted edges are corrected to be vertical. Comparing the overlay images we further see how the deviation from ground truth (shown in blue) is much smaller after going through our pipeline. Most of the remaining deviation is in areas which are not visible in the RS frame but in the GS frame (e.g behind corners).

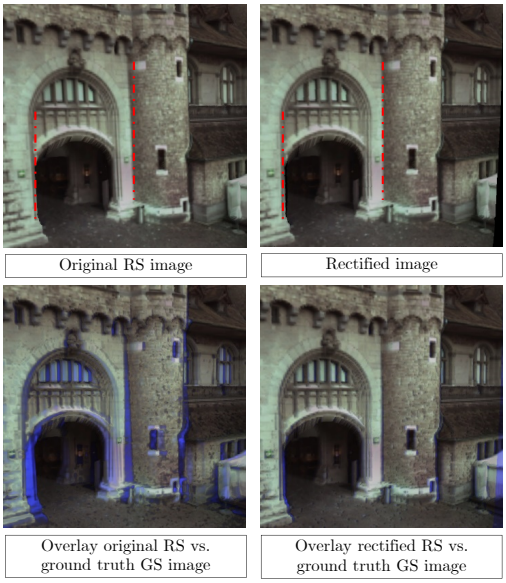

Figure 4. Results on synthetic data: examples of the RS distortions and their rectifications are marked in red. The overlay images combine two images: the original GS image and the deviation dyed in blue. The deviation is calculated as the difference between the GS image and the original or rectified RS image respectively.

5.4. Parameter Sweeps on Synthetic Data

We compare the effect of certain model parameters with and without refinement in the following evaluations. Each plot shows the mean of 50 evaluations with 50 RANSAC trials, using a threshold of 0.01 on the normalized image plane. We avoid forward motion in all experiments since it is well know that this is challenging for SfM, even for traditional GS cameras. We evaluate the 3D error as average Euclidean distance $\frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{X}_{\text{Est}}^{(i)}-\mathbf{X}_{\text{GT}}^{(i)}\right\|_2^2$ between the estimated and ground truth (GT) 3D points.

In figure 5 (left) we show the effect of the acceleration factor k. Assuming GS, we see an increasing error as k rises. We further see how the constant velocity method cannot account for the acceleration and shows an increasing error too. Finally, the error is constant assuming constant acceleration, indicating the method successfully accounts for the acceleration. Without refinement the standard deviation is significantly higher when assuming constant acceleration due to the additional estimation of the parameter k. Since the GS error increases rapidly for $k \neq 0$, we use $k = 0$ for the following evaluations. Looking at (26), we see that our method can not be used in the constant acceleration case if $k = 0$. Thus, we do not plot these results.

In figure 5 (middle) we see the influence of the readout time ratio γ as the RS effect becomes dominant (i.e. $\gamma \to 1$). Using the GS assumption the error increases, meaning the RS distortion gets larger as γ rises. In contrast, the error is constant assuming constant velocity.

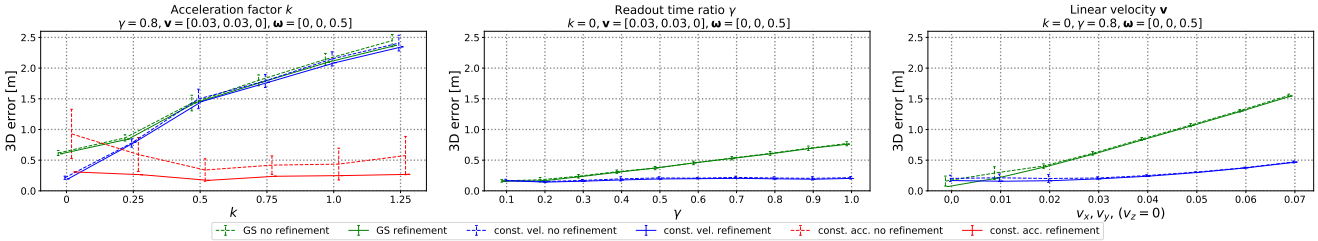

Figure 5. Sweep over acceleration factor k, readout time ratio γ and linear velocity $\mathbf{v}$. We plot the 3D error of 50 evaluations and the standard deviation among the evaluations for a 600×600 synthetic images. The respective motion parameters are depicted in the plots.

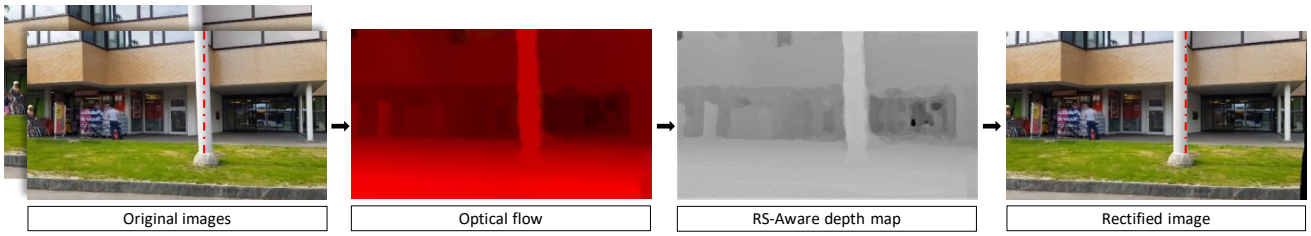

Figure 6. Result using two frames taken at Bucheggplatz in Zürich: the RS distortion seen on the slanted post is successfully rectified.

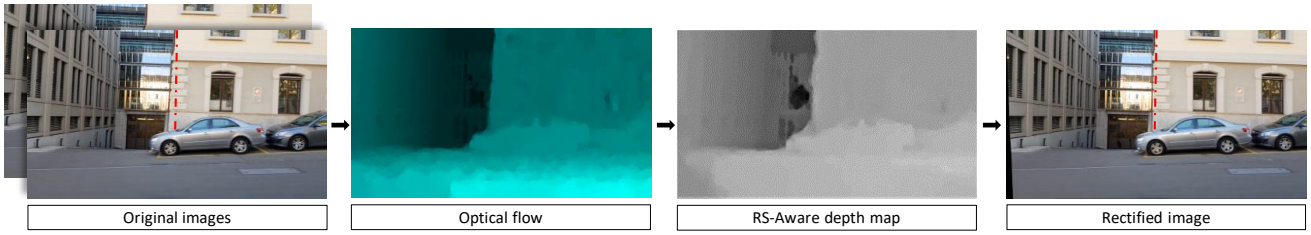

Figure 7. Result using two frames taken at ETH Zürich: the RS distortion seen on the cricked building edges is successfully rectified.

In figure 5 (right) we show the 3D error for varying linear velocities $\mathbf{v}$. We can again see that the constant velocity assumption has significantly less error than the GS case.

We further tested the method for different angular velocities $\boldsymbol{\omega}$. However, for larger angular velocities the Minimal Solver estimated a linear velocity with a high forward motion v_z instead of the correct angular velocity (despite good rectification). We assume this problem is partly caused by the simplified motion model and the degeneracy of RS SfM described by Albl *et al*. [3].

5.5. Performance on Real World Data

The results for real world images are shown in figure 6 and 7. Looking at the red lines we observe successful rectification of the RS distortion. For the method to work well, an accurate optical flow estimation is crucial. This was achieved best for uniform translations in scenes with clear and varying depth structure, few reflections and descriptive texture. In figure 7 one can see that even if the depth map is not perfect, the rectification still provides accurate results. For full HD frames running on a Laptop, the algorithm approximately requires the following computation times: flow estimate 30s, one RANSAC trial 20s and final refinement 30s. This is nowhere close to real time, limiting the possible applications.

6. Discussion

The experiments show that the accuracy of the method depends on the specific motion of the camera and the scene. As shown in section 5.4 and figure 5, the RS-aware approach can reduce the 3D error when using images from RS cameras. However, the quality of the results strongly depends on the quality of the optical flow. Since this remains a challenging task, not every setting is suited for the method described in this paper. Especially in scenes with little texture (e.g. white walls) or with complex shapes (e.g. trees), the optical flow fails to provide accurate results. Furthermore, for strong camera rotations the relative pose estimation is inaccurate.

For future work it would be interesting to extend the algorithm to sequences of more than two frames. This could be used for example to correct the RS effect in not only single images, but whole videos.

Acknowledgements

We want to thank our supervisor Dr. Oliver Saurer for several productive meetings and his helpful and immediate advice.

References

[1] S. Agarwal, K. Mierle, and Others. Ceres Solver. http://ceres-solver.org.

[2] C. Albl, Z. Kukelova, and T. Pajdla. R6P - Rolling shutter absolute pose problem. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 07-12-June, pages 2292–2300, 2015.

[3] C. Albl, A. Sugimoto, and T. Pajdla. Degeneracies in Rolling Shutter SfM. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 36–51, Cham, 2016. Springer International Publishing.

[4] J. Blanco. A tutorial on se (3) transformation parameterizations and on-manifold optimization. *University of Malaga, Tech. Rep*, (3):1–56, 2014.

[5] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[6] Y. Dai, H. Li, and L. Kneip. Rolling Shutter Camera Relative Pose: Generalized Epipolar Geometry. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4132–4140, 2016.

[7] J. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys. Building rome on a cloudless day. In *Computer Vision, ECCV 2010 - 11th European Conference on Computer Vision, Proceedings*, volume 6314 LNCS of *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pages 368–381, 11 2010.

[8] G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[9] G. Hanning, N. Forslow, P.-E. Forssen, E. Ringaby, D. Tornqvist, and J. Callmer. Stabilizing cell phone video using inertial measurement sensors. *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1–8, 2011.

[10] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[11] T. Hassner. Viewing real-world faces in 3D. In *International Conference on Computer Vision (ICCV)*, Dec. 2013.

[12] A. D. Jepson and D. J. Heeger. Linear subspace methods for recovering translation direction. *Spatial Vision in Humans and Robots*, (April 1992):in press, 1992.

[13] A. Karpenko, D. Jacobs, J. Baek, and M. Levoy. Digital Video Stabilization and Rolling Shutter Correction using Gyroscopes. *Stanford Tech Report CTSR*, pages 1–7, 2011.

[14] Y. Lao, O. Ait-Aider, and H. Araujo. Robustified Structure from Motion with rolling-shutter camera using straightness constraint. *Pattern Recognition Letters*, 111:1–8, 2018.

[15] Y. I. Ma, J. Košecká, and S. Sastry. Linear differential algorithm for motion recovery: a geometric approach. *International Journal of Computer Vision*, 36(1):71–89, 2000.

[16] M. Meilland, T. Drummond, and A. I. Comport. A Unified Rolling Shutter and Motion Blur Model for 3D Visual Registration. In *2013 IEEE International Conference on Computer Vision*, pages 2016–2023, 2013.

[17] R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*, volume 29. 1994.

[18] P. Purkait and C. Zach. Minimal Solvers for Monocular Rolling Shutter Compensation under Ackermann Motion. 2017.

[19] P. Purkait, C. Zach, and A. Leonardis. Rolling Shutter Correction in Manhattan World. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-Octob(i):882–890, 2017.

[20] O. Saurer, K. Köser, J.-Y. Bouguet, and M. Pollefeys. Rolling Shutter Stereo. In *2013 IEEE International Conference on Computer Vision*, pages 465–472, 2013.

[21] O. Saurer, M. Pollefeys, and G. H. Lee. A Minimal Solution to the Rolling Shutter Pose Estimation Problem. In *Intelligent Robots and Systems (IROS 2015), 2015 IEEE/RSJ International Conference on*, 2015.

[22] O. Saurer, M. Pollefeys, and G. H. Lee. Sparse to Dense 3D Reconstruction from Rolling Shutter Images. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3337–3345, 2016.

[23] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *2013 IEEE International Conference on Computer Vision*, pages 1385–1392, Dec 2013.

[24] B. Zhuang, L. F. Cheong, and G. H. Lee. Rolling-Shutter-Aware Differential SfM and Image Rectificatio. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 948–956, 2017.