ETH zürich

Rolling-Shutter-Aware Differential SfM and Image Rectification

Manuel Fritsche, Felix Graule, Thomas Ziegler
Supervisor: Dr. Oliver Saurer
28-May-2018

1 Setup

Given:
- Two images of a moving rolling-shutter (RS) camera (e.g. smartphone camera)
- The camera moves with linear velocity $\boldsymbol{v}$ and angular velocity $\boldsymbol{\omega}$

Goals:
- Recover the relative pose of images taken with a RS camera.
- Correct the RS effect in the images

Idea:

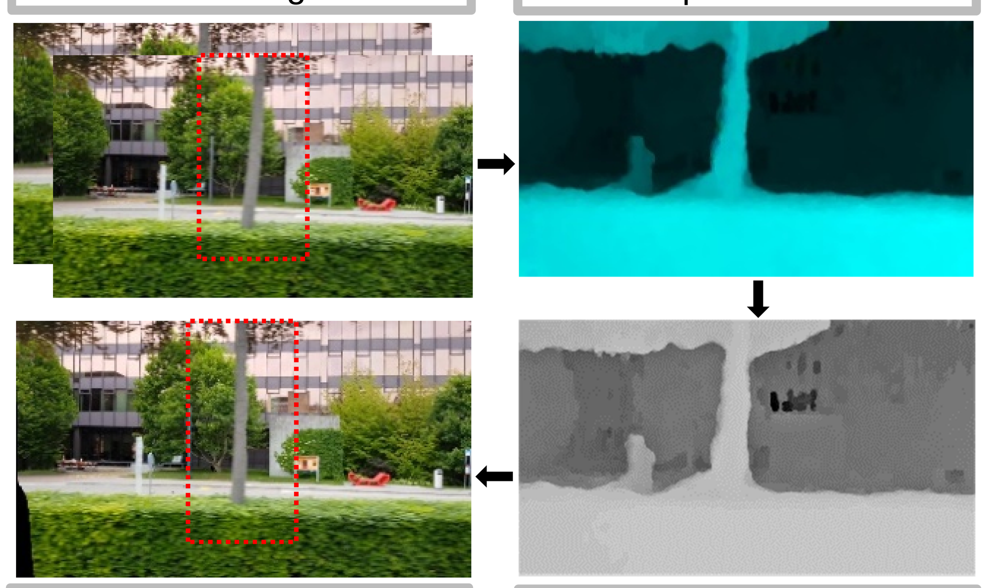

Fig. 1: Idea of the algorithm

2 RS-Aware Differential SfM

Consider image points $\boldsymbol{x}$ and optical flow $\boldsymbol{u}$

Classical differential SfM:
- This approach assumes a global-shutter (GS) camera
- The differential epipolar constraint is used for computing the pose:

$$\boldsymbol{u}^T\hat{\boldsymbol{v}}\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{s}\boldsymbol{x} = 0 \text{ with } \boldsymbol{s} = \frac{1}{2}(\hat{\boldsymbol{v}}\hat{\boldsymbol{\omega}} + \hat{\boldsymbol{\omega}}\hat{\boldsymbol{v}})$$

RS-Aware Approach [1]:

Correct $\boldsymbol{u}$ of each image row i separately by a factor β_i for the RS effect:

$$\frac{\boldsymbol{u}^T}{\beta_i}\hat{\boldsymbol{v}}\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{s}\boldsymbol{x} = 0$$

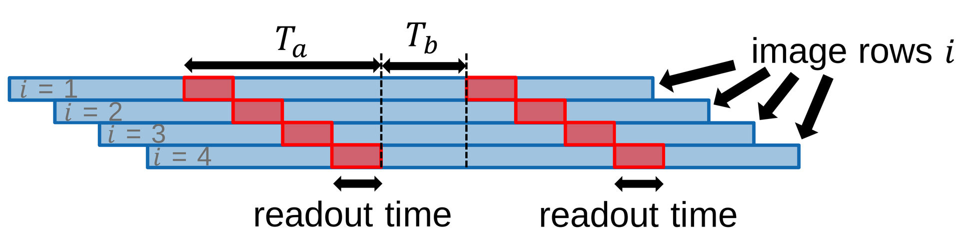

Fig. 2: Shows how the pixels of a RS camera are recorded. The image rows are taken sequentially.

Readout time ratio: $\gamma = T_a/(T_a + T_b)$

Assumptions for camera movement:
- *Constant velocity:* $\beta_i(\gamma, \boldsymbol{u})$ is a linear interpolation between scanlines
- *Constant acceleration:* $\beta_i(\gamma, \boldsymbol{u}, k)$ also depends on the acceleration factor k

References

[1] Zhuang et al., "Rolling-Shutter-Aware Differential SfM and Image Rectification", ICCV 2017

[2] Y. Ma, J. Košecká and S. Sastry, "Linear differential algorithm for motion recovery: A geometric approach", International Journal of Computer Vision

[3] Sameer Agarwal and Keir Mierle, "Ceres Solver: Tutorial & Reference", Google Inc

[4] P. Weinzaepfel, J. Revaud, Z. Harchaoui and C. Schmid, "Deepflow: Large displacement optical flow with deep matching", ICCV 2013

[5] T. Hassner, "Viewing real-world faces in 3d. In International Conference on Computer Vision", ICCV 2013

[6] O. Saurer, K. Koser, J.-Y. Bouguet and M. Pollefeys, "Rolling shutter stereo. In International Conference on Computer Vision", ICCV 2013

3 Implementation

- Synthetic data: calculate ground truth
- Real world data: use Deepflow [4]

Optical Flow

- Use 9 image points to build the matrix $\boldsymbol{Z}(k)$ with $\boldsymbol{Z}(k)\boldsymbol{e} = 0$ from the differential epipolar constraint. $\boldsymbol{e}$ is the vector of unknowns in the constraint.
- Solve for k by using $\det(\boldsymbol{Z}(k)) = 0$.
- Calculate $\boldsymbol{v}$ and $\boldsymbol{\omega}$ from $\boldsymbol{Z}(k)$ by following the approach in [2].

RANSAC: Minimal Solver → Depth Estimator

- Sample 9 image points for each trial (without replacement).
- Apply Minimal Solver to selected points and then use Depth Estimator on all points.
- Estimate optical flow from $\boldsymbol{v}, \boldsymbol{\omega}, \beta_i$ and depth values.
- Choose inliers by using the error of the calculated optical flow.

- Calculate the depth values $\boldsymbol{x}^{(z)}$ by using Ceres Solver [3] to optimize the estimated optical flow:

$$\underset{\boldsymbol{x}^{(z)}}{\operatorname{argmin}} \sum_i \left\| \boldsymbol{u}_i - \boldsymbol{u}_i^{\text{est}}\left(x_i^{(z)}, \boldsymbol{v}, \boldsymbol{\omega}, k\right)\right\|^2$$

Nonlinear Refinement

- Refines the final RANSAC result
- Optimize the estimated optical flow using Ceres Solver [3] :

$$\underset{\boldsymbol{x}^{(z)}, \boldsymbol{v}, \boldsymbol{\omega}, k}{\operatorname{argmin}} \sum_i \left\| \boldsymbol{u}_i - \boldsymbol{u}_i^{\text{est}}\left(x_i^{(z)}, \boldsymbol{v}, \boldsymbol{\omega}, k\right)\right\|^2$$

Image Rectification

- Back-project each pixel on each scanline into the 3D space which results in a point cloud.
- Project the point cloud onto the image plane using the pose corresponding to the first scanline.

Fig. 3: Shows the pipeline of the algorithm. The implementation is following [1].

4 Experiments on Synthetic Data

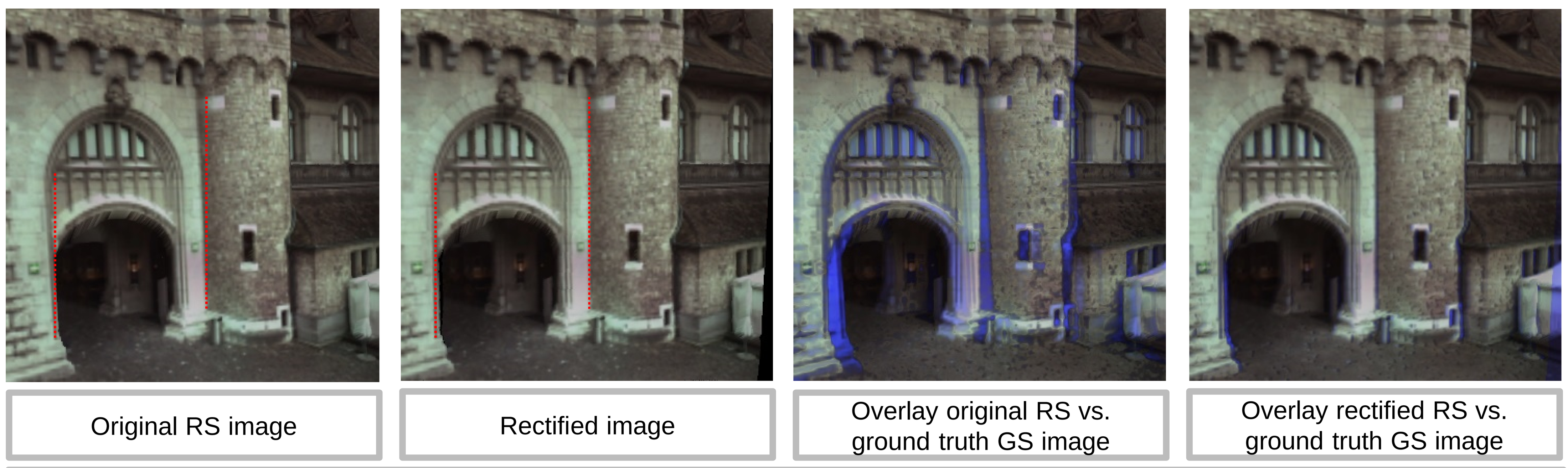

Fig. 4: The above "castle" images were rendered using [5] based on a 3D model provided by [6]. Overlay deviation is shown in blue.

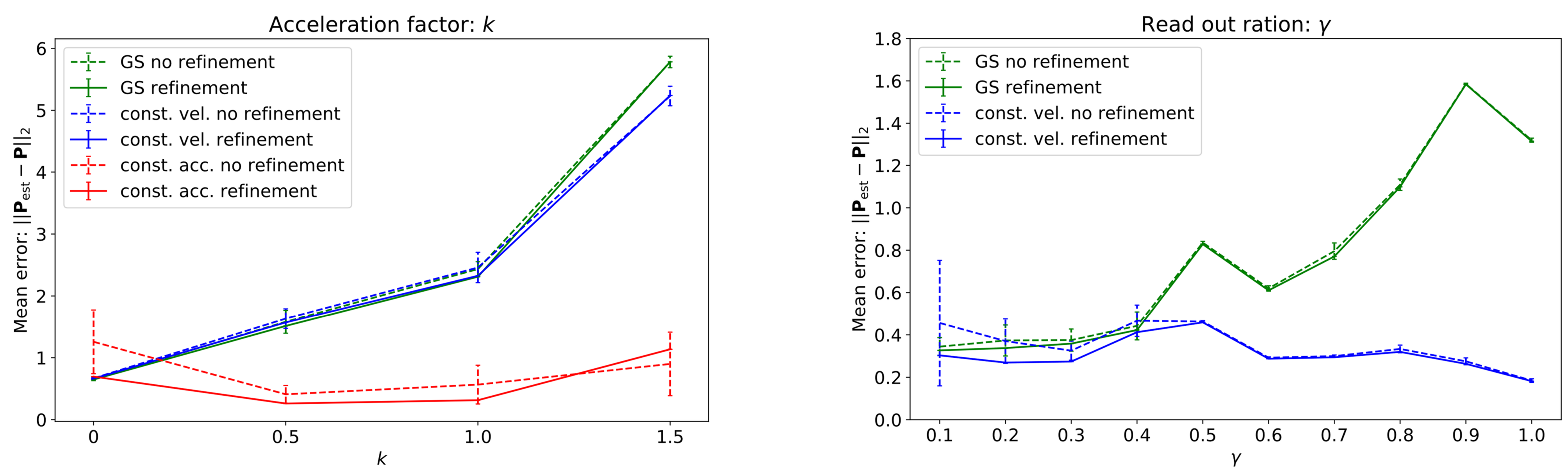

Fig. 5: Mean (and standard deviation) of the error of estimated 3D points vs. ground truth for different values of k (left) or γ (right) using rendered images of the "castle" model above. The mean error was estimated over 20 evaluations with 30 RANSAC trials each.

5 Experiments on Real World Data

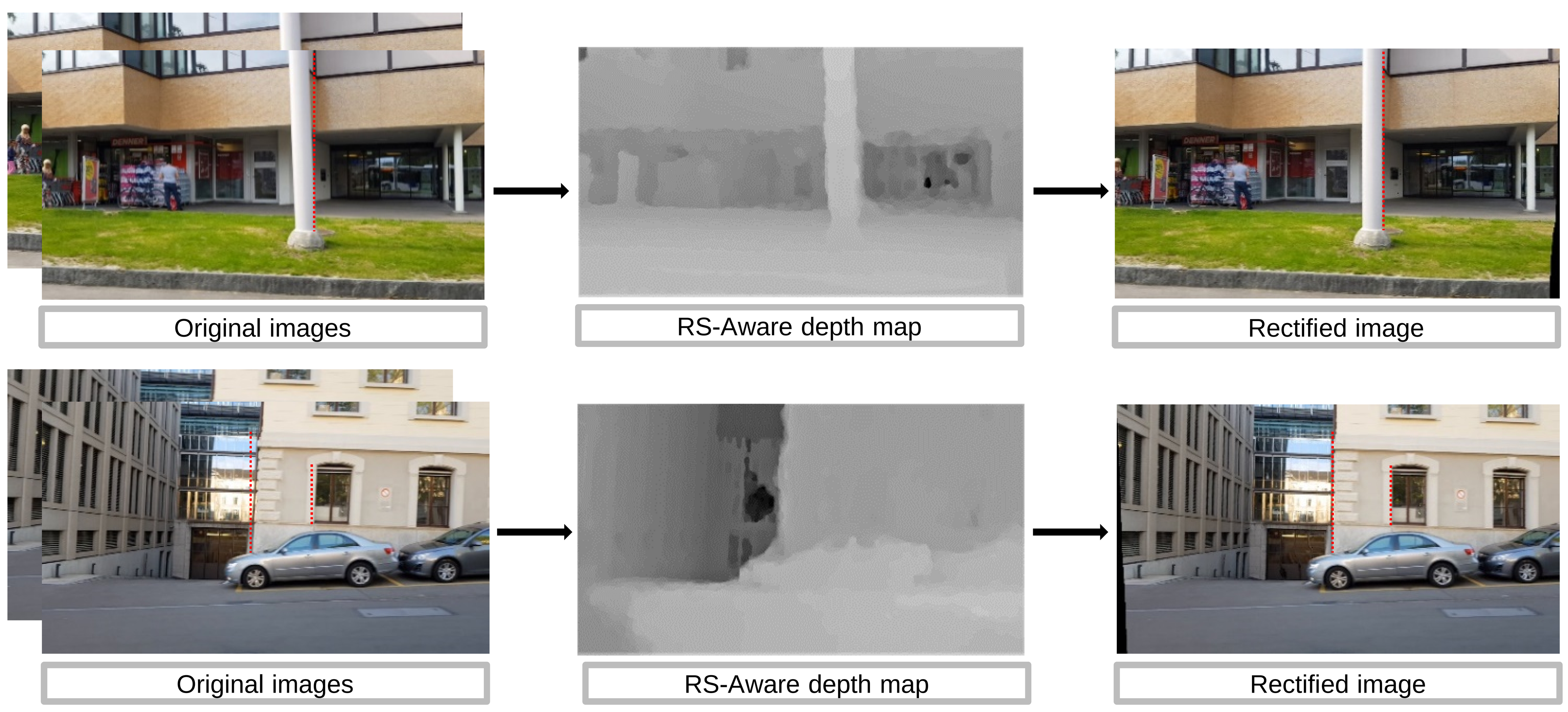

Fig. 6: Images taken from the real world. The optical flow is calculated by using Deepflow [4].