# Minimal Rolling Shutter Absolute Pose with Unknown Focal Length and Radial Distortion - Supplementary Material

Zuzana Kukelova[1], Cenek Albl[2], Akihiro Sugimoto[3], Konrad Schindler[2], and Tomas Pajdla[4]

[1] VRG, Faculty of Electrical Engineering, Czech Technical University in Prague
[2] Photogrammetry and Remote Sensing, ETH Zurich, Switzerland
[3] National Institute of Informatics, Tokyo, Japan
[4] Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague

## 1 Additional synthetic experiments

In this section we show additional evaluation of the proposed algorithms on synthetic data. We show five more experiments demonstrating the practical advantages of using R7Pf and R7Pfr.

In the first two experiments we gradually increased rotational and translational velocity to the same values as in the experiments in Figure 2 in the main paper, e.g. rotation velocity up to 30 degrees/frame and relative translational velocity up to 1/10 of the camera distance from the scene per the duration of a frame. The difference from the experiments in the main paper is that this time the camera orientation is not set to identity and, therefore, R7Pf and R7Pfr have to be initialized by an initial rotation. We use the output rotation from P4Pf and P4Pfr to initialize R7Pf and R7Pfr respectively.

The data in the first experiment was generated without radial distortion whereas in the second experiment we used a fixed radial distortion of about half the maximum value of the one used in experiment in Figure 3 in the main paper. Figure 1 shows how R7P and R7Pfr should behave in a practical scenario with a moderate RS distortion, unknown focal length and no radial distortion and Figure 2 shows the case for radial distortion.

We can see that without radial distortion, the initialization by both P4Pf and P4Pfr is good enough to ensure R7Pf and R7Pfr provide a significantly better camera pose and focal length than the existing solutions. The P4Pfr+R7Pfr is significantly less stable on data without radial distortion, which indicates that the RS effect is being explained partially by the radial distortion. As expected on data with radial distoriton, P4Pf+R7Pf performs significantly poorer which indicates that radial distortion present in the image is being explained by some RS distortions, similar effect as with R7Pfr on non-distorted data. This is also visible in the extremely poor result of P4Pf+R6P. R7Pf initialized by P4Pfr on average outperforms P4Pfr+R6P, but it is clear from the results of both, that the radial distortion estimated by a solver without RS model (P4Pfr) is poor. R7Pfr provides the best performance and significantly outperforms all alternatives.

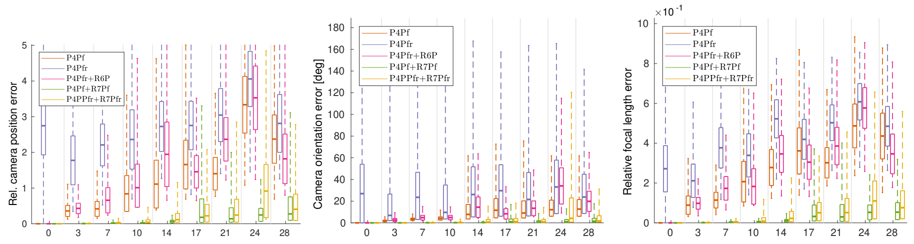

**Fig. 1.** Performance of R7Pf and R7Pfr on data with increasing RS distortion and unknown focal length, when the initial orientation is initialized by P4Pf and P4Pfr respectively.

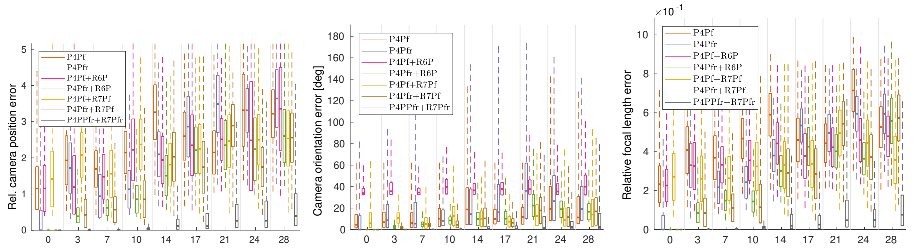

**Fig. 2.** Performance of R7Pf and R7Pfr on data with increasing RS distortion, moderate radial distortion and unknown focal length, when the initial orientation is initialized by P4Pf and P4Pfr respectively.

Further, we tested the effect of orientation initialization in isolation. We created synthetic data with increasing distance of the initial camera orientation from the true orientation. The RS motion was of medium magnitude, the focal length unknown and the radial distortion was kept at zero to give both solvers a fair comparison. As you can see in Figure 3 both R7Pf and R7Pfr are affected by the distance of the linearization point from the true orientation. However, even under significant initialization error (30 degrees), they still provide very good results and outperform the best alternative, which is the P4Pf followed by R6P.

Another interesting factor is the number of iterations our solvers require to converge. We tested this on synthetic data again. This time, we used the same data as in Figure 2 of the main paper, where RS motion was increased gradually. We collected the number of iterations which the solvers needed to reach certain values of algebraic error. The algebraic error can be used as one of the possible criteria of convergence as it is related to how well are the original equations satisfied. Figure 4 shows the histograms of numbers of iterations for each solver using three reasonable stopping criteria values. Both R7Pf and R7Pfr converge in majority of cases under 10 iterations even for the most strict threshold. For a more loose threshold of $10^{-4}$, R7Pfr converges under 5 iterations in 90% of cases. R7Pfr shows overall faster convergence rate than R7Pf.

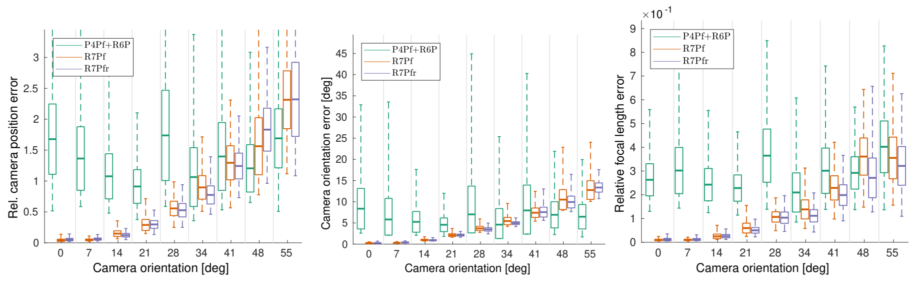

**Fig. 3.** Increasing angular distance from the linearization point of the camera orientation, i.e. how far the initial orientation estimate is. One can see that R7Pf and R7Pfr are quite robust to the initial orientation estimate. Even with an extremely bad initialization with 30 degrees error they still outperform the combination of P4Pf+R6P.

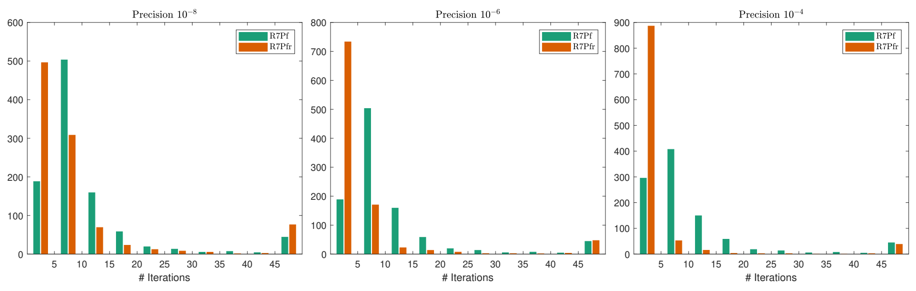

**Fig. 4.** Convergence analysis of R7Pf and R7Pfr. Histogram showing number of iterations before the algorithms converged to algebraic error smaller than $10^{-8}$ (left), $10^{-6}$ (middle) and $10^{-4}$ (right). Both R7Pf and R7Pfr tend to converge under 10 iterations in the majority of cases, but R7Pfr usually converges faster.

The runtimes of our non-optimized Matlab implementations of our solvers [5] are 120us for R7Pf and 400us for R7Pfr per iteration. Based on the runtimes of the computationally heavy parts (matrix inversion, eigen-decomposition) we expect a reasonable implementation in C to take 60us per iteration of R7Pf and 100us per iteration of R7Pfr. Depending on the accuracy requirements, 2-5 iterations are needed.

As a last synthetic data experiment we investigated the behavior of both presented solvers on planar data. It is known that R6P provides inferior results in cases where all six input 3D points lie on a single plane. We verified whether this is the case for R7Pf and R7Pfr as well. We used the same settings as in the experiment from Figure 2 in the main paper (increasing RS motion), but this time with a planar scene. As is seen from the results in Figure 5, both solvers are affected by the planarity of the 3D points. However, the results are still significantly better than those of their non-RS counterparts - P4Pf and P4Pfr.

[5] The implementation of our solvers is available at *github.com/CenekAlbl/RnP*.

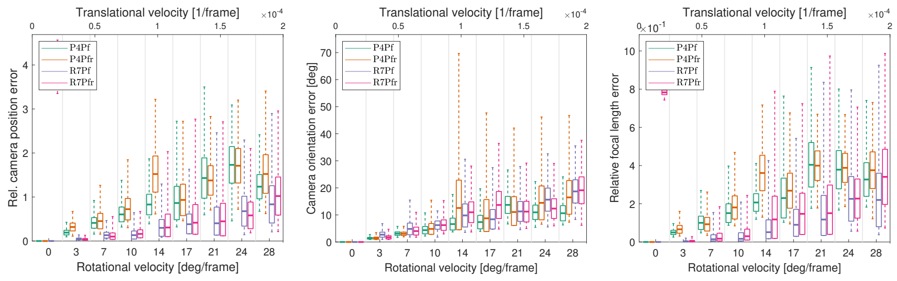

**Fig. 5.** Analysis of behavior on planar scenes. With the same settings as in the experiment in Figure 2 of the main paper (increasing RS distortion) R7Pf and R7Pfr both peform worse on planar scenes, but still significantly better than P4Pf and P4Pfr.

## 2 Qualitative trajectory evaluation

Here we show an example of the camera poses obtained by the compared algorithms. Figure 6 shows the camera centers calculated by P4Pfr+R7Pfr (cyan), P4Pfr+R7Pfr+LO (green), P4Pfr+R6P (blue) and P4Pfr+R6P+LO (red) connected by lines which form a continuous trajectory of a drone performing a fast maneuver (right) and a rollercoaster performing a helix motion (left). One can observe that our solutions provide significantly more stable pose especially during fast motions. The baseline algorithms are prone to providing completely wrong pose at multiple occasions and overall suffer from lower accuracy caused by the lower number of detected inliers as well as interplay of the RS and radial distortion parameters.

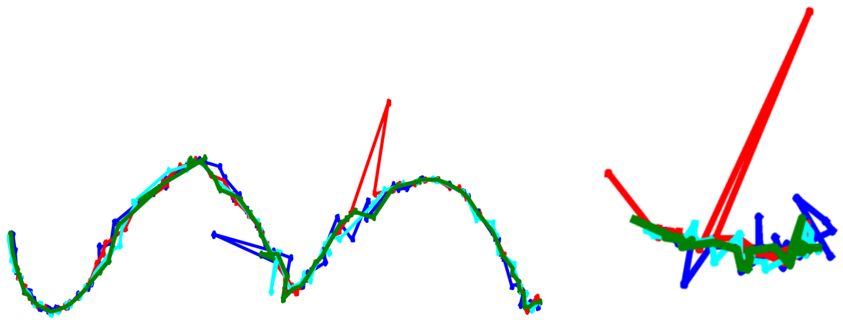

**Fig. 6.** Reconstructed camera trajectories of the Gopro rollercoaster dataset (top) and Gopro drone 2 (bottom). The P4Pfr+R7Pfr (cyan) and P4Pfr+R7Pfr+LO (green) provide much more stable camera path than P4Pfr+R6P (blue) and P4Pfr+R6P+LO (red) in the critical places where camera motion is high.