Linear Solution to the Minimal Absolute Pose Rolling Shutter Problem

Zuzana Kukelova[1(✉)], Cenek Albl[2], Akihiro Sugimoto[3], and Tomas Pajdla[2]

[1] Visual Recognition Group (VRG), FEE, Czech Technical University in Prague, Prague, Czech Republic
kukelzuz@fel.cvut.cz
[2] Czech Institute of Informatics, Robotics and Cybernetics (CIIRC), Czech Technical University in Prague, Prague, Czech Republic
[3] National Institute of Informatics, Tokyo, Japan

Abstract. This paper presents new efficient solutions to the rolling shutter camera absolute pose problem. Unlike the state-of-the-art polynomial solvers, we approach the problem using simple and fast linear solvers in an iterative scheme. We present several solutions based on fixing different sets of variables and investigate the performance of them thoroughly. We design a new alternation strategy that estimates all parameters in each iteration linearly by fixing just the non-linear terms. Our best 6-point solver, based on the new alternation technique, shows an identical or even better performance than the state-of-the-art R6P solver and is two orders of magnitude faster. In addition, a linear non-iterative solver is presented that requires a non-minimal number of 9 correspondences but provides even better results than the state-of-the-art R6P. Moreover, all proposed linear solvers provide a single solution while the state-of-the-art R6P provides up to 20 solutions which have to be pruned by expensive verification.

Keywords: Rolling shutter · Absolute pose · Minimal solvers

1 Introduction

Rolling shutter (RS) cameras are omnipresent. They can be found in smartphones, consumer, professional, and action cameras and even in self-driving cars. RS cameras are cheaper, and easier to produce, than global shutter cameras. They also posses other advantages over the global shutter cameras, such as higher achievable frame-rate or longer exposure times.

This work was supported by the European Regional Development Fund under the project IMPACT (reg. no. CZ.02.1.01/0.0/0.0/15_003/0000468), EC H2020-ICT-731970 LADIO project, ESI Fund, OP RDE programme under the project International Mobility of Researchers MSCA-IF at CTU No. CZ.02.2.69/0.0/0.0/17_050/0008025, and Grant-in-Aid for Scientific Research (Grant No. 16H02851) of the Ministry of Education, Culture, Sports, Science and Technology of Japan. A part of this work was done when Zuzana Kukelova was visiting the National Institute of Informatics (NII), Japan, funded in part by the NII MOU grant.

C. V. Jawahar et al. (Eds.): ACCV 2018, LNCS 11363, pp. 265–280, 2019.
https://doi.org/10.1007/978-3-030-20893-6_17

There is, however, a significant drawback when using them for computer vision applications. When the scene or the camera is moving during image capture, images produced by RS cameras will become distorted. The amount and type of distortion depends on the type and speed of camera motion and on the depth of the scene. It has been shown that RS image distortions can cause problems for standard computer vision methods such as Structure from Motion [1], visual SLAM [2] or multi-view dense stereo [3]. Therefore, having a special camera model for rolling shutter cameras is desirable.

The camera absolute pose computation is a fundamental problem in many computer vision tasks such as Structure from Motion, augmented reality, visual SLAM, and visual localization. The problem is to compute the camera pose from 3D points in the world and their 2D projections into an image. The minimal number of correspondences necessary to solve the absolute pose problem for a perspective calibrated camera is three. The first solution to this problem was introduced by Grunert [4] and since then it was many times revisited [5–7]. Other work has focused on computing the absolute pose from a larger than the minimal number of correspondences [8–12]. All of the previous work consider a perspective camera model, which is not suitable for dynamic RS cameras.

Recently, as RS cameras became more and more common, the focus turned to computing camera absolute pose from images containing RS effects. First, several RS camera motion models were introduced in [13]. A solution to RS absolute pose using non-minimal (eight and half) number of points was presented in [14]. It relied on a non-linear optimization and required a planar scene.

In [15], video sequences were exploited and the absolute camera pose was computed sequentially using a non-linear optimization starting from the previous camera pose. Another approach using video sequences was used for visual SLAM in [2] where the camera motion estimated from previous frames was used to compensate the RS distortion in the next frame prior to the optimization.

A polynomial solution that is globally optimal was presented in [16]. It uses Gloptipoly [17] solver to find a solution from 7 or more points. Authors show that the method provides better results than [14], but the runtime is in the order of seconds, making it impractical for typical applications such as RANSAC.

The first minimal solution to the rolling shutter camera absolute pose problem was presented in [18]. It uses the minimal number of six 2D to 3D point correspondences and the Gröbner basis method to generate an efficient solver. The proposed R6P is based on the constant linear and angular velocity model as in [1,14,16] but it uses the first order approximation to both the camera orientation and angular velocity, and, therefore, it requires an initialization of the camera orientation, e.g., from P3P [7]. Paper [18] has shown that R6P solver significantly outperforms perspective P3P solver in terms of camera pose precision and the number of inliers captured in the RANSAC loop.

1.1 Motivation

It has been demonstrated in the literature that RS camera absolute pose is beneficial and often necessary when dealing with RS images from moving camera

or dynamic scene. Still, until now, all the presented solutions have significant drawbacks that make them impractical for general use.

The state-of-the-art solutions require a non-minimal or a larger number of points [14,16], planar scene [14], video sequences [1,2,15], are very slow [16] and provide too many solutions [18].

If one requires a practical algorithm similar to P3P, but working on RS images, the closest method available is R6P [18]. However, R6P still needs around 1.7 ms to compute the camera pose, compared to around 3 µs for P3P. Therefore, in typical applications where P3P is used, one would suffer a several orders of magnitude slowdown compared to P3P. This makes it hard to use for real-time applications such as augmented reality. In addition, R6P provides up to 20 real solutions, which need to be verified. This makes tasks like RANSAC, which uses hundreds or thousands of iterations and verifies all solutions, extremely slow compared to P3P. This motivates us to create a solution to RS absolute pose problem with similar performance to R6P [18] and runtime comparable to P3P.

1.2 Contribution

In this work we present solutions that remove previously mentioned drawbacks of the state-of-the-art methods and provide practical and fast rolling shutter camera absolute pose solvers. We take a different approach to formulating the problem and propose linear solutions to rolling shutter camera absolute pose. Specifically, we present the following RS absolute camera pose solvers:

- a 6-point linear iterative solver, which provides identical or even better solutions than R6P in 10 µs, which is up to 170× faster than R6P. This solver is based on a new alternating method;
- two 6-point linear iterative solvers that outperform R6P for purely translational motion;
- a 9-point linear non-iterative solver that provides more accurate camera pose estimates than R6P in 20 µs;

All solvers are easy to implement and they return a single solution. We formulate the problem of RS camera absolute pose in Sect. 2. Derivations of all new solvers are in Sect. 3. Section 4 contains experiments verifying the feasibility of the proposed solvers and it compares them against P3P and R6P [18].

2 Problem Formulation

For calibrated perspective cameras, the projection equation can be written as

$$\lambda_i \mathbf{x}_i = \mathtt{R}\mathbf{X}_i + \mathtt{C}, \tag{1}$$

where $\mathtt{R}$ and $\mathtt{C}$ are the rotation and translation bringing a 3D point $\mathbf{X}_i$ from a world coordinate system to the camera coordinate system with $\mathbf{x}_i = [r_i, c_i, 1]^\top$, and scalar $\lambda_i \in \mathbb{R}$. For RS cameras, every image row is captured at different

time and hence at a different position when the camera is moving during the image capture. Camera rotation $\mathtt{R}$ and translation $\mathtt{C}$ are therefore functions of the image row r_i being captured

$$\lambda_i \mathtt{x}_i = \lambda_i \begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix} = \mathtt{R}(r_i)\mathtt{X}_i + \mathtt{C}(r_i). \tag{2}$$

In recent work [1,3,13,14,16,18], it was shown that for the short time-span of a frame capture, the camera translation $\mathtt{C}(r_i)$ can be approximated with a simple constant velocity model as

$$\mathtt{C}(r_i) = \mathtt{C} + (r_i - r_0)\mathtt{t}, \tag{3}$$

where $\mathtt{C}$ is the camera center corresponding to the perspective case, i.e. when $r_i = r_0$, and $\mathtt{t}$ is the translational velocity.

The camera rotation $\mathtt{R}(r_i)$ can be decomposed into two rotations to represent the camera initial orientation by $\mathtt{R}_\mathtt{v}$ and the change of orientation during frame capture by $\mathtt{R}_\mathtt{w}(r_i - r_0)$.

In [16,18], it was observed that it is usually sufficient to linearize $\mathtt{R}_\mathtt{w}(r_i - r_0)$ around the initial rotation $\mathtt{R}_\mathtt{v}$ using the first order Taylor expansion such that

$$\lambda_i \begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix} = (\mathtt{I} + (r_i - r_0)[\mathtt{w}]_\times)\,\mathtt{R}_\mathtt{v}\mathtt{X}_i + \mathtt{C} + (r_i - r_0)\mathtt{t}, \tag{4}$$

where $[\mathtt{w}]_\times$ is a skew-symmetric matrix of vector $\mathtt{w}$. The model (4), with linearized rolling shutter rotation, will deviate from the reality with increasing rolling shutter effect. Still, it is usually sufficient for most of the rolling shutter effects present in real situations.

In [18], a linear approximation to the camera orientation $\mathtt{R}_\mathtt{v}$ was used to solve the rolling shutter absolute pose problem from a minimal number of six 2D-3D point correspondences. This model has the form

$$\lambda_i \begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix} = (\mathtt{I} + (r_i - r_0)[\mathtt{w}]_\times)\,(\mathtt{I} + [\mathtt{v}]_\times)\,\mathtt{X}_i + \mathtt{C} + (r_i - r_0)\mathtt{t}. \tag{5}$$

The drawback of the model (5) is that $\mathtt{R}_\mathtt{v}$ is often not small and thus cannot be linearized. Therefore, the accuracy of the model is dependent on the initial orientation of the camera in the world frame. In [18], it was shown that the standard P3P algorithm [7] is able to estimate camera orientation with sufficient precision even for high camera rotation velocity and therefore P3P can be used to bring the camera rotation matrix $\mathtt{R}_\mathtt{v}$ close to the identity, where (5) works reasonably.

The model (5) leads to a system of six quadratic equations in six unknowns. This system has 20 solutions and it was solved in [18] using the Gröbner basis method [19,20]. The Gröbner basis solver [18] for the R6P rolling shutter problem

requires the G-J elimination of a 196×216 matrix and computing the eigenvectors of a 20×20 matrix. The R6P solver runs for about 1.7 ms and thus is too slow in many practical situations.

We will next show how to simplify this model by linearizing Eq. (5) and yet still obtaining a similar performance as the Gröbner basis R6P absolute pose solver [18] for the original model (5).

3 Linear Rolling Shutter Solvers

We present here several linear iterative solvers to the minimal absolute pose rolling shutter problem. All these solvers start with the model (5) and they use six 2D-3D image point correspondences to estimate 12 unknowns $\mathtt{v}, \mathtt{C}, \mathtt{w}$, and $\mathtt{t}$. The proposed solvers differ in the way how the system (5) is linearized. Additionally we propose a linear non-iterative 9 point absolute pose rolling shutter solver.

3.1 R6P$^{\mathtt{w,t}}_{\mathtt{v,C}}$ Solver

The R6P$^{\mathtt{w,t}}_{\mathtt{v,C}}$ solver is based on the idea of alternating between two linear solvers. The first R6P$_{\mathtt{v,C}}$ solver fixes the rolling shutter parameters $\mathtt{w}$ and $\mathtt{t}$ in (5) and estimates only the camera parameters $\mathtt{v}$ and $\mathtt{C}$. The second R6P$_{\mathtt{w,t}}$ solver fixes the camera parameters $\mathtt{v}$ and $\mathtt{C}$ and estimates only the rolling shutter parameters $\mathtt{w}$ and $\mathtt{t}$. Both these partial solvers results in 12 linear equations in 6 unknowns that can be solved in the least square sense. The motivation for this solver comes from the fact that even for larger rolling shutter speed, the camera parameters $\mathtt{v}$ and $\mathtt{C}$ can be estimated quite accurately.

The R6P$^{\mathtt{w,t}}_{\mathtt{v,C}}$ solver starts with $\mathtt{w}_0 = \mathtt{0}$ and $\mathtt{t}_0 = \mathtt{0}$ and, in the first iteration, uses linear R6P$_{\mathtt{v,C}}$ solver to estimate $\mathtt{v}_1$ and $\mathtt{C}_1$. Using the estimated $\mathtt{v}_1$ and $\mathtt{C}_1$, the linear solver R6P$_{\mathtt{w,t}}$ estimates $\mathtt{w}_1$ and $\mathtt{t}_1$. This process is repeated until the desired precision is obtained or a maximum number of iterations is reached.

The R6P$^{\mathtt{w,t}}_{\mathtt{v,C}}$ solver does not perform very well in our experiments, which we account to the fact that it never estimates the pose parameters $\mathtt{v}$,$\mathtt{C}$ and the motion parameters $\mathtt{w}$,$\mathtt{t}$ together in one step. Nevertheless, we present this solver as a logical first step when considering the iterative approach to RS absolute pose problem.

3.2 R6P$^{\mathtt{w}}_{\mathtt{v,C,t}}$ and R6P$^{\mathtt{w,t}}_{\mathtt{v,C,t}}$ Solver

To avoid problems of the R6P$^{\mathtt{w,t}}_{\mathtt{v,C}}$ solver, we introduce the R6P$^{\mathtt{w}}_{\mathtt{v,C,t}}$ solver. The R6P$^{\mathtt{w}}_{\mathtt{v,C,t}}$ solver alternates between two solvers, i.e. the linear R6P$_{\mathtt{v,C,t}}$ solver, which fixes only the rolling shutter rotation $\mathtt{w}$ and estimates $\mathtt{v}, \mathtt{C}$ and $\mathtt{t}$, and the R6P$_{\mathtt{w}}$ solver that estimates only the rolling shutter rotation $\mathtt{w}$ using the fixed $\mathtt{v}, \mathtt{C}$ and $\mathtt{t}$. The R6P$_{\mathtt{v,C,t}}$ solver solves 12 linear equations in 9 unknowns and the R6P$_{\mathtt{w}}$ solver solves 12 linear equations in 3 unknowns in the least square sense. Since the first R6P$_{\mathtt{v,C,t}}$ solver assumes unknown rolling shutter translation, the camera parameters are estimated with better precision than in the case of the

$\text{R6P}_{\mathtt{v},\mathtt{C}}$ solver. Moreover, in many applications, e.g. cameras on a car, cameras often undergo only a translation motion, and therefore $\mathtt{w}$ is negligible. In such situations, the first iteration of the $\text{R6P}_{\mathtt{v},\mathtt{C},\mathtt{t}}$ solver already provides very precise estimates of the camera parameters.

Another approach is to use only the $\mathtt{v}$ and $\mathtt{C}$ estimated by $\text{R6P}_{\mathtt{v},\mathtt{C},\mathtt{t}}$ solver and in the second step re-estimate the rolling shutter translation $\mathtt{t}$ together with the rolling shutter rotation $\mathtt{w}$ using the linear $\text{R6P}_{\mathtt{w},\mathtt{t}}$ solver. The solver based on this strategy will be referred to as $\text{R6P}^{\mathtt{w},\mathtt{t}}_{\mathtt{v},\mathtt{C},\mathtt{t}}$.

The resulting solvers $\text{R6P}^{\mathtt{w}}_{\mathtt{v},\mathtt{C},\mathtt{t}}$ and $\text{R6P}^{\mathtt{w},\mathtt{t}}_{\mathtt{v},\mathtt{C},\mathtt{t}}$, again, alternate between the two linear solvers until the desired precision is obtained or a maximum number of iterations is reached. We show in the experiments that those solvers outperform R6P in the case of pure translational motion.

3.3 $\text{R6P}^{[\mathtt{v}]_\times}_{\mathtt{v},\mathtt{C},\mathtt{w},\mathtt{t}}$ Solver

The $\text{R6P}^{[\mathtt{v}]_\times}_{\mathtt{v},\mathtt{C},\mathtt{w},\mathtt{t}}$ solver estimates all unknown parameters $\mathtt{v}, \mathtt{C}, \mathtt{w}$ and $\mathtt{t}$ together in one step. To avoid non-linearity in (5), the solver fixes $[\mathtt{v}]_\times$ that appears in the nonlinear term $[\mathtt{w}]_\times[\mathtt{v}]_\times$ in (5). Thus the solver solves equations

$$\lambda_i \begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix} = (\mathtt{I} + (r_i - r_0)[\mathtt{w}]_\times)\,\mathtt{X}_i + [\mathtt{v}]_\times \mathtt{X}_i + (r_i - r_0)[\mathtt{w}]_\times[\hat{\mathtt{v}}]_\times \mathtt{X}_i + \mathtt{C} + (r_i - r_0)\mathtt{t}, \quad (6)$$

where $\hat{\mathtt{v}}$ is a fixed vector.

In the first iteration $\hat{\mathtt{v}}$, is set to the zero vector and the term $(r_i - r_0)[\mathtt{w}]_\times[\hat{\mathtt{v}}]_\times \mathtt{X}_i$ in (6) disappears. This is usually a sufficient approximation. The explanation for this is as follows. After the initialization with P3P the camera rotation is already close to the identity and in real applications the rolling shutter rotation $\mathtt{w}$ during the capture is usually small. Therefore, the nonlinear term $[\mathtt{w}]_\times[\mathtt{v}]_\times$ is small, sometimes even negligible, and thus it can be considered to be zero in the first iteration.

In the remaining iterations we fix $\hat{\mathtt{v}}$ in the $(r_i - r_0)[\mathtt{w}]_\times[\hat{\mathtt{v}}]_\times \mathtt{X}_i$ term to be equal to the $\mathtt{v}_i$ estimated in the previous iteration of the $\text{R6P}^{[\mathtt{v}]_\times}_{\mathtt{v},\mathtt{C},\mathtt{w},\mathtt{t}}$ solver. Note that we fix only $\mathtt{v}$ that appears in the nonlinear term $[\mathtt{w}]_\times[\mathtt{v}]_\times$ and there is still another term with $\mathtt{v}$ in (6) from which a new $\mathtt{v}$ can be estimated. Therefore, all parameters are estimated at each step which is a novel alternating strategy. To our knowledge, all existing algorithms that are based on the alternating optimization approach completely fix a subset of the variables, meaning that they cannot estimate all the variables in one step.

The $\text{R6P}^{[\mathtt{v}]_\times}_{\mathtt{v},\mathtt{C},\mathtt{w},\mathtt{t}}$ in each iteration solves only one system of 12 linear equations in 12 unknowns and is therefore very efficient. In experiments we will show that the $\text{R6P}^{[\mathtt{v}]_\times}_{\mathtt{v},\mathtt{C},\mathtt{w},\mathtt{t}}$ provides very precise estimates already after 1 iteration and after 5 iterations it has virtually the same performance as the state-of-the-art R6P solver [18].

3.4 R9P

Our final solver is a non-iterative solver that uses a non-minimal number of nine 2D-3D point correspondences. We note that the projection Eq. (6) can be rewritten as

$$\lambda_i \begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix} = (\mathtt{I} + [\mathtt{v}]_\times)\,\mathtt{X}_i + \mathtt{C} + (r_i - r_0)([\mathtt{w}]_\times(\mathtt{I} + [\hat{\mathtt{v}}]_\times)\mathtt{X}_i + \mathtt{t}). \tag{7}$$

We can substitute the term $[\mathtt{w}]_\times(\mathtt{I}+[\hat{\mathtt{v}}]_\times)$ in (7) with a 3×3 unknown matrix $\mathtt{R}_{\mathrm{RS}}$. After eliminating the scalar values λ_i by multiplying Eq. (7) from the left by the skew symmetric matrix for vector $\begin{bmatrix} r_i & c_i & 1 \end{bmatrix}^\top$ and without considering the internal structure of the matrix $\mathtt{R}_{\mathrm{RS}}$, we obtain three linear equations in 18 unknowns, i.e. $\mathtt{v}, \mathtt{C}, \mathtt{t}$, and 9 unknowns in $\mathtt{R}_{\mathrm{RS}}$. Since only two from these tree equations are linearly independent we need nine 2D-3D point correspondences to solve this problem.

Note that the original formulation (5) was an approximation to the real rolling shutter camera model and therefore the formulation with a general 3×3 matrix $\mathtt{R}_{\mathrm{RS}}$ is yet a different approximation to this model.

4 Experiments

We tested the proposed solvers on a variety of synthetic and real datasets and compared the results with the original R6P solver [18] as well as P3P. We followed the general pattern of experiments used in [18] in order to provide consistent comparison on the additional factor of experiments that are specific to our iterative solvers such as their convergence.

To analyze the accuracy of the estimated camera poses and velocities, we used synthetic data in the following setup. A random set of 3D points was generated in a cubic region with $x, y, z \in [-1; 1]$ and a camera with a distance $d \in [2; 3]$ from the origin and pointing towards the 3D points. The camera was set to be calibrated, i.e. $\mathtt{K} = \mathtt{I}$ and the field of view was set to 45°. Rolling shutter projections were created using a constant linear velocity and a constant angular velocity with various magnitudes.

Using the constant angular velocity model for generating the data ensures that our data is not generated with the same model as the one that is estimated by the solvers (linear approximation to a rotation). Although the used model is just an approximation of the real rolling shutter model and we could have chosen another one, e.g. constant angular acceleration, we consider the constant angular velocity model as a reasonable description of the camera motion during the short time period of frame capture.

We used 6 points for the original R6P and all proposed R6P iterative solvers. In order to provide P3P with the same data, we used all possible triplets from the 6 points used by R6P and then chose the best result. For R9P we used 9 points. Unless stated otherwise, all iterative solvers were run for maximum 5 iterations in the experiments.

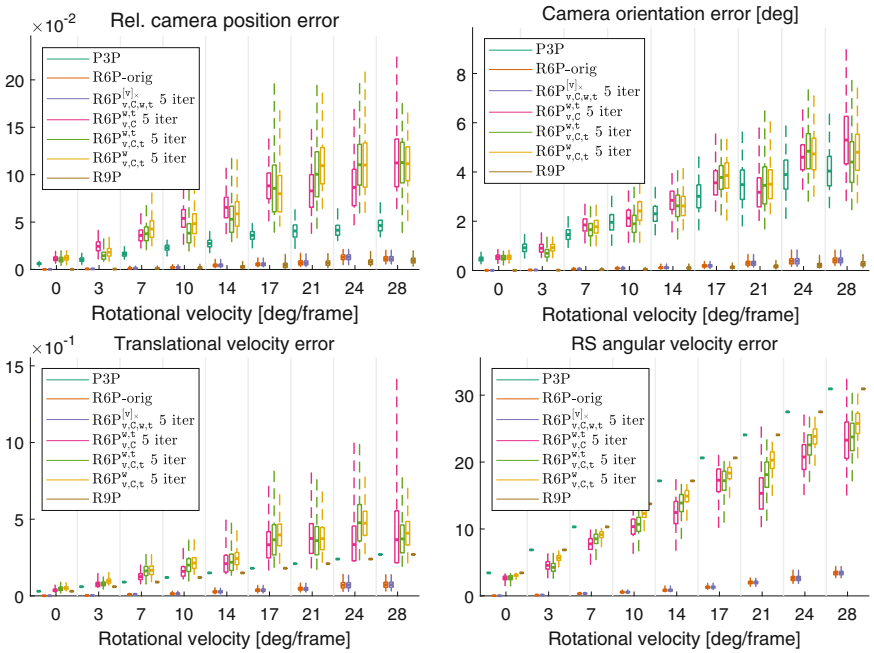

Fig. 1. Experiment on synthetic data focusing on the precision of estimated camera poses and velocities. Notice that the performance of $\mathrm{R6P}^{[\mathtt{v}]_\times}_{\mathtt{v,C,w,t}}$ is identical to R6P. In terms of camera pose these two solvers are slightly outperformed by R9P. Other linear solvers perform very poorly in all respects.

4.1 Synthetic Data

In the first experiment, we gradually increased the camera velocities during capture. The maximum translational velocity was 0.3 per frame and the maximum angular velocity was 30° per frame. Figure 1 shows the results, from which we can see how the increasing RS deformation affects the estimated camera pose and also estimated camera velocities in those solvers.

Rotational and Translational Motion: In agreement with [18], R6P provides much better results than P3P thanks to the RS camera model. The newly proposed solver $\mathrm{R6P}^{[\mathtt{v}]_\times}_{\mathtt{v,C,w,t}}$ provides almost identical results to R6P at much lower computation cost (cf. Table 1). The best estimates of the camera pose are provided by R9P at the cost of using more than minimal number of points. The other 6-point iterative solutions are performing really bad, often providing worse results than P3P. In the next experiment we tested the sensitivity of the proposed solvers to increasing levels of image noise. Figure 2 right shows that the new solvers have approximately the same noise sensitivity as R6P [18].

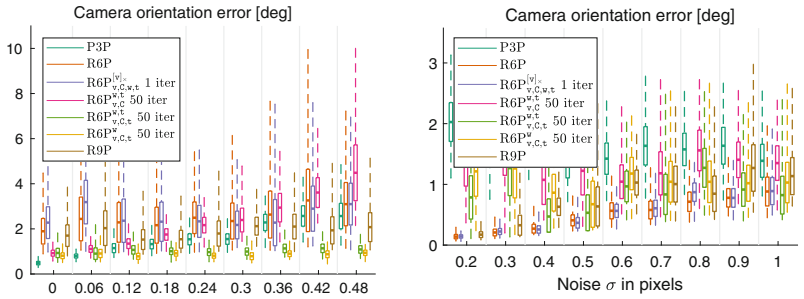

Fig. 2. (Left) Purely translational camera motion, increasing on the x axis. Image noise with σ 1pix. Notice that $\mathrm{R6P}^{\mathtt{w}}_{\mathtt{v,C,t}}$ and $\mathrm{R6P}^{\mathtt{w,t}}_{\mathtt{v,C,t}}$ now outperform all the others. (Right) Performance on general camera motion with increasing image noise.

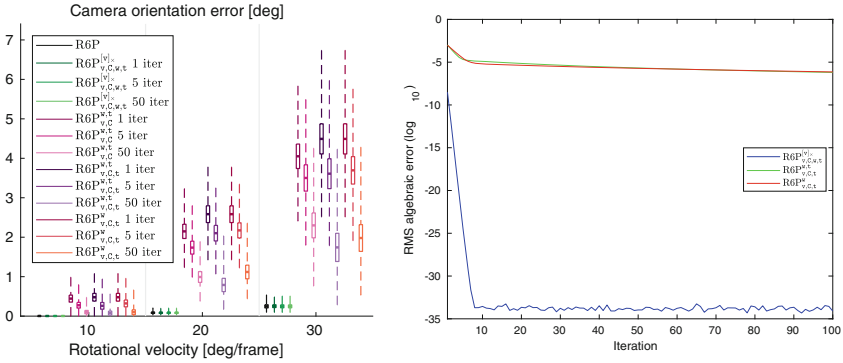

Fig. 3. Testing the convergence of the iterative solvers. All iterative solvers have been run with 1, 5 and 50 iterations on data with $\mathtt{R} = \mathtt{I}$ and increasing RS effect (left). Convergence of the algebraic error using the three viable iterative solvers (right).

Translational Motion Only: The advantage of solvers $\mathrm{R6P}^{\mathtt{w,t}}_{\mathtt{v,C,t}}$ and $\mathrm{R6P}^{\mathtt{w}}_{\mathtt{v,C,t}}$ is when the motion of the camera is purely translational, or close to it, which is a common scenario in, e.g., a moving car or a moving train. In such cases, both original R6P and $\mathrm{R6P}^{[\mathtt{v}]\times}_{\mathtt{v,C,w,t}}$ provide significantly worse estimates of the camera pose. We explain this by the fact that $\mathrm{R6P}^{\mathtt{w,t}}_{\mathtt{v,C,t}}$ and $\mathrm{R6P}^{\mathtt{w}}_{\mathtt{v,C,t}}$ are constrained to estimate only camera translation in the initial step, whereas R6P and $\mathrm{R6P}^{[\mathtt{v}]\times}_{\mathtt{v,C,w,t}}$ try to explain the image noise by the camera rotation. See Fig. 2 left. This fact can be used to create a "joined solver" that runs both $\mathrm{R6P}^{\mathtt{w}}_{\mathtt{v,C,t}}$ and $\mathrm{R6P}^{[\mathtt{v}]\times}_{\mathtt{v,C,w,t}}$ and gives better performance than R6P [18] while still being significantly faster.

Convergence: For $\mathrm{R6P}^{\mathtt{w,t}}_{\mathtt{v,C}}$, $\mathrm{R6P}^{\mathtt{w}}_{\mathtt{v,C,t}}$, and $\mathrm{R6P}^{\mathtt{w,t}}_{\mathtt{v,C,t}}$, the maximum 5 iterations might not be enough to converge to a good solution, whereas $\mathrm{R6P}^{[\mathtt{v}]\times}_{\mathtt{v,C,w,t}}$ seems to perform at its best. We thus increased the maximum number of iterations. Figure 3 (left) shows that the performance of $\mathrm{R6P}^{\mathtt{w,t}}_{\mathtt{v,C}}$, $\mathrm{R6P}^{\mathtt{w}}_{\mathtt{v,C,t}}$, and $\mathrm{R6P}^{\mathtt{w,t}}_{\mathtt{v,C,t}}$ is

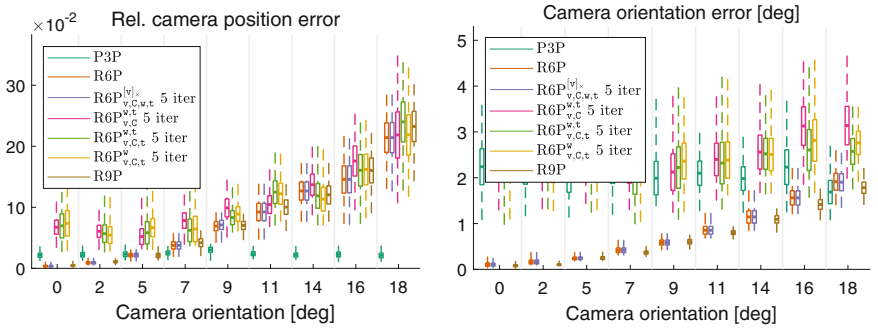

Fig. 4. Experiment showing the effect of the linearized camera pose which is present in all models. The further the camera orientation is from the linearization point, the worse are the results. $\mathrm{R6P}^{[\mathtt{v}]_\times}_{\mathtt{v,C,w,t}}$ matches the results of R6P and so does R9P.

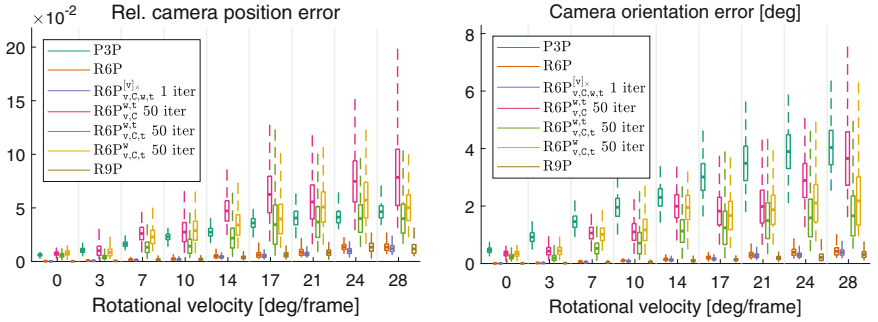

Fig. 5. Increasing the camera motion and estimating camera pose with all solvers being initialized with P3P. $\mathrm{R6P}^{[\mathtt{v}]_\times}_{\mathtt{v,C,w,t}}$ and R9P now provide consistently excellent results, comparable or outperforming those of R6P at a fraction of the computation cost. $\mathrm{R6P}^{\mathtt{w,t}}_{\mathtt{v,C}}$, $\mathrm{R6P}^{\mathtt{w}}_{\mathtt{v,C,t}}$ and $\mathrm{R6P}^{\mathtt{w,t}}_{\mathtt{v,C,t}}$ with 50 iterations now perform better than P3P, but still not as good as the other RS solvers.

improved by increasing the maximum number of iterations to 50. However, it is still far below the performance of R6P, $\mathrm{R6P}^{[\mathtt{v}]_\times}_{\mathtt{v,C,w,t}}$, and R9P. $\mathrm{R6P}^{[\mathtt{v}]_\times}_{\mathtt{v,C,w,t}}$ performs as well as the R6P even with a single iteration, making it two orders of magnitude faster alternative. The algebraic error, evaluated on the Eq. (5), of the three viable solvers converges within 8 steps on average, see Fig. 3 (right).

The Effect of Linearized Camera Rotation Model: Since all the proposed solvers have a linearized form of the camera orientation, in the same way as R6P [18], we tested how being further from the linearization point affects the performance (Fig. 4). The camera orientation was set to be at a certain angle from $\mathtt{R} = \mathtt{I}$. The camera velocities were set to 0.15 per frame for the translation and 15° per frame for the rotation. In [18] the authors show that R6P outperforms P3P in terms of camera center estimation up to 6° away from the initial

Table 1. Average timings on 2.5 GHz i7 CPU per iteration for all used solvers.

Solver	P3P	R6P	$\text{R6P}^{[\mathtt{v}]\times}_{\mathtt{v,C,w,t}}$	$\text{R6P}^{\mathtt{w}}_{\mathtt{v,C,t}}$	$\text{R6P}^{\mathtt{w,t}}_{\mathtt{v,C,t}}$	$\text{R6P}^{\mathtt{w,t}}_{\mathtt{v,C}}$	R9P
Time per iteration	3 µs	1700 µs	10 µs	24 µs	30 µs	27 µs	20 µs
max # of solutions	4	20	1	1	1	1	1

$\mathtt{R}$ estimate and up to 15° away from $\mathtt{R}$ for the camera orientation estimate. Our results in Fig. 4 show similar behavior and identical results of R6P and $\text{R6P}^{[\mathtt{v}]\times}_{\mathtt{v,C,w,t}}$. R9P performs comparable to both, even slightly outperforming them in terms of camera orientation estimation.

Using P3P as Initial Estimate: Last synthetic experiment shows the performance of the solvers when using the initial estimate of $\mathtt{R}$ from the result of P3P. The camera orientation was randomly generated and the camera motion was increased as in the first experiment. P3P was computed first and the 3D scene was pre-rotated using $\mathtt{R}$ from P3P. This shows probably the most practical usage among all R6P solvers. To make the figure more informative, we chose the number of iterations for $\text{R6P}^{\mathtt{w,t}}_{\mathtt{v,C}}$, $\text{R6P}^{\mathtt{w}}_{\mathtt{v,C,t}}$, and $\text{R6P}^{\mathtt{w,t}}_{\mathtt{v,C,t}}$ to be 50 as the 5 iterations already proved to be insufficient, see Fig. 1. We also set the maximum number of iterations for $\text{R6P}^{[\mathtt{v}]\times}_{\mathtt{v,C,w,t}}$ to 1, to demonstrate the potential of this solver.

As seen in Fig. 5, $\text{R6P}^{[\mathtt{v}]\times}_{\mathtt{v,C,w,t}}$ provides at least as good, or even better, results than R6P after only a single iteration. This is a significant achievement since the computational cost of $\text{R6P}^{[\mathtt{v}]\times}_{\mathtt{v,C,w,t}}$ is two orders of magnitude less than of R6P. With 50 iterations the other iterative solvers perform better than P3P, but considering the computational cost of 50 iterations, which is even higher than that of a R6P, we cannot recommend using them in such a scenario.

Computation Time: The computation times for all the tested solvers are shown in Table 1. One iteration of $\text{R6P}^{[\mathtt{v}]\times}_{\mathtt{v,C,w,t}}$ is two orders of magnitude faster than R6P. According to the experiments, even one iteration of $\text{R6P}^{[\mathtt{v}]\times}_{\mathtt{v,C,w,t}}$ provides very good results, comparable with R6P and 5 iterations always match the results of R6P or even outperform them at 34× the speed. Note that R9P can be even faster than $\text{R6P}^{[\mathtt{v}]\times}_{\mathtt{v,C,w,t}}$ because it is non-iterative and runs only once and is therefore as fast as 2 iterations of $\text{R6P}^{[\mathtt{v}]\times}_{\mathtt{v,C,w,t}}$. One iteration of $\text{R6P}^{\mathtt{w,t}}_{\mathtt{v,C}}$, $\text{R6P}^{\mathtt{w}}_{\mathtt{v,C,t}}$ and $\text{R6P}^{\mathtt{w,t}}_{\mathtt{v,C,t}}$ is around three times slower than $\text{R6P}^{[\mathtt{v}]\times}_{\mathtt{v,C,w,t}}$ but still almost two orders of magnitude faster than R6P.

4.2 Real Data

We used the publicly available datasets from [1] and we show the results of the same frames shown in [18] (seq1, seq8, seq20 and seq22) in order to make a relevant comparison. We also added one more real dataset (House), containing high RS effects from a fast moving drone carrying a GoPro camera. The 3D-2D correspondences were obtained in the same way as in [18] by reconstructing the

scene using global shutter images and then matching the 2D features from the RS images to the reconstructed 3D points.

We performed RANSAC with 1000 iterations for each solver to estimate the camera pose and calculated the number of inliers. The inlier threshold was set to 2 pixels in the case of the data from [1] which was captured by handheld iPhone at 720p and to 8 pixels for the GoPro footage which was recorder in 1080p. The higher threshold in the second case allowed to capture a reasonable number of inliers even for such fast camera motions. The results in Fig. 6 show the number of inliers captures over the sequences of images. We see that the performance of $\mathrm{R6P}^{[\mathtt{v}]_\times}_{\mathtt{v},\mathtt{C},\mathtt{w},\mathtt{t}}$ with 5 iterations is virtually identical to R6P. The results of $\mathrm{R6P}^{\mathtt{w},\mathtt{t}}_{\mathtt{v},\mathtt{C},\mathtt{t}}$ and $\mathrm{R6P}^{\mathtt{w}}_{\mathtt{v},\mathtt{C},\mathtt{t}}$ are also very similar and often outperform R6P and $\mathrm{R6P}^{[\mathtt{v}]_\times}_{\mathtt{v},\mathtt{C},\mathtt{w},\mathtt{t}}$, except for the most challenging images in the House dataset.

The performance of $\mathrm{R6P}^{\mathtt{w},\mathtt{t}}_{\mathtt{v},\mathtt{C}}$ is unstable, sometimes performing comparable to or below P3P. In seq20 in particular, there is almost exclusively a fast translational camera motion. The drop in performance can therefore be explained by $\mathrm{R6P}^{\mathtt{w},\mathtt{t}}_{\mathtt{v},\mathtt{C}}$ being the only solver that does not estimate the translational velocity $\mathtt{t}$ in the first step. R9P performs solidly across all the experiments and on the most challenging House dataset it even provides significantly better results.

To test another useful case of camera absolute pose, which is augmented reality, we created an environment filled with Aruco [21] markers in known positions. We set up the markers in such a way that they covered three perpendicular walls. The scene was recorded with a camera performing translational and rotational motion, similar to what a human does when looking around or shaking the head.

All solvers were used in RANSAC with 100 iterations to allow some robustness to outliers and noise. Note that 100 iterations of RANSAC would take at least 200 ms for R6P excluding the inlier verification. That makes R6P not valuable for real time purposes (in practice only less than 10 iterations of R6P would give realtime performance). On the other hand, 100 runs of $\mathrm{R6P}^{[\mathtt{v}]_\times}_{\mathtt{v},\mathtt{C},\mathtt{w},\mathtt{t}}$ with 5 iterations take around 5 ms (200 fps) and $\mathrm{R6P}^{\mathtt{w}}_{\mathtt{v},\mathtt{C},\mathtt{t}}$ takes around 12.5 ms (80 fps). We did not test solvers $\mathrm{R6P}^{\mathtt{w},\mathtt{t}}_{\mathtt{v},\mathtt{C}}$, $\mathrm{R6P}^{\mathtt{w},\mathtt{t}}_{\mathtt{v},\mathtt{C},\mathtt{t}}$ and R9P in this experiment. This is because the performance of $\mathrm{R6P}^{\mathtt{w},\mathtt{t}}_{\mathtt{v},\mathtt{C}}$ is unstable, the performance of $\mathrm{R6P}^{\mathtt{w},\mathtt{t}}_{\mathtt{v},\mathtt{C},\mathtt{t}}$ is almost identical to $\mathrm{R6P}^{\mathtt{w}}_{\mathtt{v},\mathtt{C},\mathtt{t}}$ and with R9P we do not have a way to extract the camera motion parameters and the reprojection without these parameters does not provide fair comparison.

We evaluated the reprojection error in each frame on all the detected markers. The results are shown in Fig. 7. All the rolling shutter solvers outperform P3P in terms of precision of the reprojections. $\mathrm{R6P}^{[\mathtt{v}]_\times}_{\mathtt{v},\mathtt{C},\mathtt{w},\mathtt{t}}$ again provides identical performance to R6P. $\mathrm{R6P}^{\mathtt{w}}_{\mathtt{v},\mathtt{C},\mathtt{t}}$ has a slight edge over the others, which is interesting, considering its poor performance on the synthetic data.

Figure 7 gives a visualization of the estimated camera pose by reprojecting a cube in front of the camera. There is a significant misalignment between the cube and the scene during camera motion when using P3P pose estimate. In comparison, all the rolling shutter solvers keep the cube much more consistent with respect to the scene.

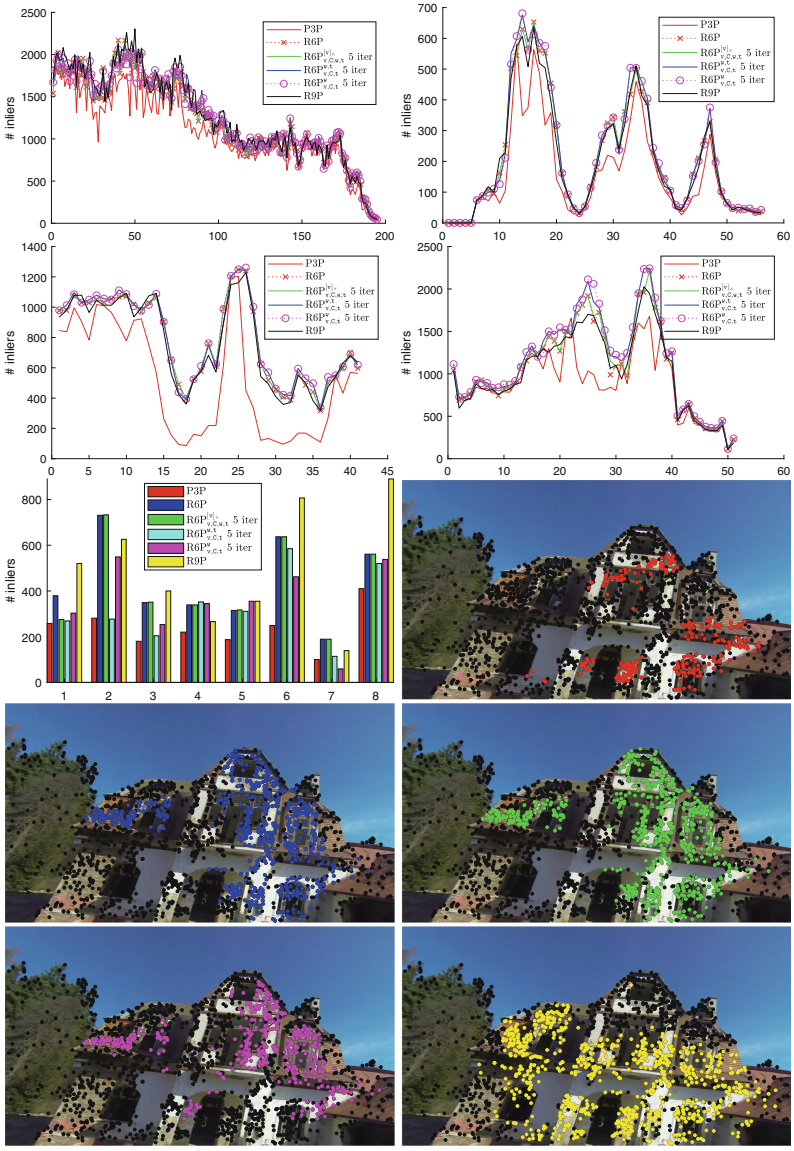

Fig. 6. Number of inliers on real data sequences. From top to bottom, left to right: seq01, seq08, seq20, seq22 and House. The x axis contains frame numbers. The bar graph for the House figure is used because there is no temporal relationship between adjacent frames so a line graph does not make sense. Following are sample images from the House dataset frame 6, containing a high amount of RS distortion. In this frame, R9P provided significantly more inliers than other methods. The results of R6P and $\mathrm{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{C},\mathbf{w},\mathbf{t}}$ are again identical, with the small exception of the first frame. The colored inliers in the sample images follow the same colors of algorithms as in the bar graph for House sequence. (Color figure online)

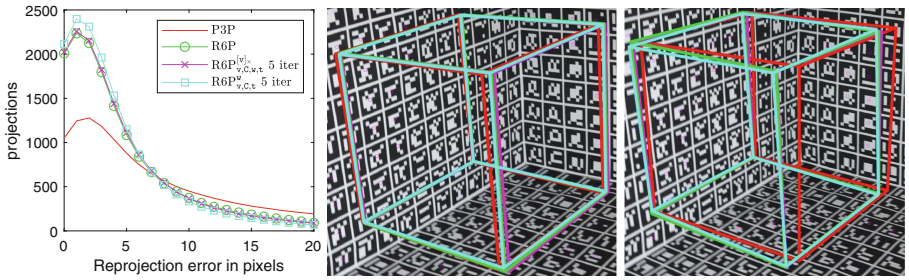

Fig. 7. Histogram of reprojection errors on the Aruco markers in the augmented reality experiment. The rolling shutter absolute pose solvers (R6P in magenta, $\mathrm{R6P}^{[\mathtt{v}]\times}_{\mathtt{v,C,w,t}}$ in green, $\mathrm{R6P}^{\mathtt{w}}_{\mathtt{v,C,t}}$ in cyan) keep the cube in place during camera motion whereas P3P (red) reprojects the cube all over the place. (Color figure online)

5 Conclusions

We revisited the problem of rolling shutter camera absolute pose and proposed several new practical solutions. The solutions are based on iterative linear solvers that improve the current state-of-the-art methods in terms of speed while providing the same precision or better. The practical benefit of our solvers is also the fact that they provide only a single solution, compared to up to 20 solutions of R6P [18].

The overall best performing $\mathrm{R6P}^{[\mathtt{v}]\times}_{\mathtt{v,C,w,t}}$ solver needs only a single iteration to provide similar performance to R6P while being approximately 170x faster. At 5 iterations the performance of R6P is matched while the new $\mathrm{R6P}^{[\mathtt{v}]\times}_{\mathtt{v,C,w,t}}$ solver is still approximately 34x faster than R6P. This allows for much broader applicability, especially in the area of augmented reality, visual SLAM and other real-time applications.

We also proposed 3 other iterative linear solvers ($\mathrm{R6P}^{\mathtt{w,t}}_{\mathtt{v,C}}$, $\mathrm{R6P}^{\mathtt{w,t}}_{\mathtt{v,C,t}}$, $\mathrm{R6P}^{\mathtt{w}}_{\mathtt{v,C,t}}$) that alternate between estimating different camera pose and velocity parameters. These three solvers are slower than $\mathrm{R6P}^{[\mathtt{v}]\times}_{\mathtt{v,C,w,t}}$ but still almost two orders of magnitude faster than R6P. While not as precise as R6P or $\mathrm{R6P}^{[\mathtt{v}]\times}_{\mathtt{v,C,w,t}}$ in the synthetic experiments, they proved usefulness on the real data, providing more inliers and better reprojections than P3P and even R6P. We presented these three solvers mainly because they follow the concept of making the rolling shutter absolute pose equations linear by alternatively fixing some variables and then others. Although $\mathrm{R6P}^{\mathtt{w,t}}_{\mathtt{v,C,t}}$ and $\mathrm{R6P}^{\mathtt{w}}_{\mathtt{v,C,t}}$ do not offer the fastest and most precise results, they performed best in some of the experiments, especially for purely translational motion, and we think they are worth mentioning.

Last but not least we presented a non-iterative linear solver that uses 9 correspondences. This solver is as fast as 2 iterations of $\mathrm{R6P}^{[\mathtt{v}]\times}_{\mathtt{v,C,w,t}}$ and proved to be the most precise in terms of estimated camera pose in the synthetic experiments and provided solid performance on the real data.

Altogether, this paper presents a big step forward in practical computation of rolling shutter camera absolute pose, making it more available in real world applications.

References

1. Hedborg, J., Forssén, P.E., Felsberg, M., Ringaby, E.: Rolling shutter bundle adjustment. In: CVPR, pp. 1434–1441 (2012)
2. Klein, G., Murray, D.: Parallel tracking and mapping on a camera phone. In: IEEE ISMAR, ISMAR 2009, pp.83–86 (2009)
3. Saurer, O., Koser, K., Bouguet, J.Y., Pollefeys, M.: Rolling shutter stereo. In: ICCV, pp. 465–472 (2013)
4. Grunert, J.A.: Das Pothenotische Problem in erweiterter Gestalt nebst über seine Anwendungen in der Geodäsie (1841)
5. Haralick, R., Lee, D., Ottenburg, K., Nolle, M.: Analysis and solutions of the three point perspective pose estimation problem. In: CVPR, pp. 592–598 (1991)
6. Ameller, M.A., Triggs, B., Quan, L.: Camera pose revisited: new linear algorithms. In: 14eme Congres Francophone de Reconnaissance des Formes et Intelligence Artificielle. Paper in French 2002 (2002)
7. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**, 381–395 (1981)
8. Lepetit, V., Moreno-Noguer, F., Fua, P.: EPnP: an accurate O(n) solution to the PnP problem. Int. J. Comput. Vis. **81**, 155–166 (2009)
9. Quan, L., Lan, Z.: Linear n-point camera pose determination. IEEE PAMI **21**, 774–780 (1999)
10. Triggs, B.: Camera pose and calibration from 4 or 5 known 3D points. In: ICCV, vol. 1, pp. 278–284 (1999)
11. Wu, Y., Hu, Z.: PnP problem revisited. J. Math. Imaging Vis. **24**, 131–141 (2006)
12. Zhi, L., Tang, J.: A complete linear 4-point algorithm for camera pose determination (2002)
13. Meingast, M., Geyer, C., Sastry, S.: Geometric models of rolling-shutter cameras. Computing Research Repository abs/cs/050 (2005)
14. Ait-Aider, O., Andreff, N., Lavest, J.M., Martinet, P.: Simultaneous object pose and velocity computation using a single view from a rolling shutter camera. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3952, pp. 56–68. Springer, Heidelberg (2006). https://doi.org/10.1007/11744047_5
15. Hedborg, J., Ringaby, E., Forssen, P.E., Felsberg, M.: Structure and motion estimation from rolling shutter video. In: ICCV Workshops, pp. 17–23 (2011)
16. Magerand, L., Bartoli, A., Ait-Aider, O., Pizarro, D.: Global optimization of object pose and motion from a single rolling shutter image with automatic 2D-3D matching. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7572, pp. 456–469. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33718-5_33
17. Henrion, D., Lasserre, J.B., Lofberg, J.: GloptiPoly 3: moments, optimization and semidefinite programming. Optim. Methods Softw. **24**, 761–779 (2009)
18. Albl, C., Kukelova, Z., Pajdla, T.: R6P - rolling shutter absolute pose problem. In: CVPR, pp. 2292–2300 (2015)

19. Cox, D., Little, J., O'Shea, D.: Using Algebraic Geometry, 2nd edn. Springer, New York (2005). https://doi.org/10.1007/b138611
20. Kukelova, Z., Bujnak, M., Pajdla, T.: Automatic generator of minimal problem solvers. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008. LNCS, vol. 5304, pp. 302–315. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88690-7_23
21. Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F., Marín-Jiménez, M.: Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognit. **47**, 2280–2292 (2014)