

R6P - Rolling Shutter Absolute Pose Problem

Cenek Albl¹

Zuzana Kukelova²

Tomas Pajdla¹

¹Czech Technical University in Prague,
Faculty of Electrical engineering,
166 27 Praha 6, Technicka 2,
Czech Republic

{alblcene,pajdla}@cmp.felk.cvut.cz

²Microsoft Research Ltd,
21 Station Road,
Cambridge CB1 2FB, UK
a-zukuke@microsoft.com

Abstract

We present a minimal, non-iterative solution to the absolute pose problem for images from rolling shutter cameras. Absolute pose problem is a key problem in computer vision and rolling shutter is present in a vast majority of today's digital cameras. We propose several rolling shutter camera models and verify their feasibility for a polynomial solver. A solution based on linearized camera model is chosen and verified in several experiments. We use a linear approximation to the camera orientation, which is meaningful only around the identity rotation. We show that the standard P3P algorithm is able to estimate camera orientation within 6 degrees for camera rotation velocity as high as 30deg/frame. Therefore we can use the standard P3P algorithm to estimate camera orientation and to bring the camera rotation matrix close to the identity. Using this solution, camera position, orientation, translational velocity and angular velocity can be computed using six 2D-to-3D correspondences, with orientation error under half a degree and relative position error under 2%. A significant improvement in terms of the number of inliers in RANSAC is demonstrated.

1. Introduction

Perspective-n-point problem (PnP) for calibrated cameras is the task of finding a camera orientation and translation from n 2D-to-3D correspondences. It is a key problem in many computer vision applications such as structure from motion, camera localization, object localization and visual odometry. PnP has been thoroughly studied in the past with first solution being published in 1841 by Grunert and later reviewed in [5]. The PnP problem for calibrated cameras can be formulated as a system of simple polynomial equations and solved from three points in a closed form. Many authors focused on different formulations of the problem, comparing numerical stability, speed or methods how to

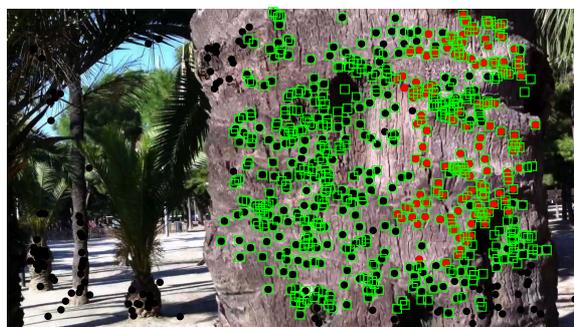


Figure 1: Result of a standard P3P and our R6P algorithm applied on image with rolling shutter distortion. All tentative 2D-3D correspondences are shown in black, inliers found by P3P are red and inliers found by R6P are in green squares. Notice that R6P found many more matches than P3P.

calculate the camera pose from more than three correspondences [2, 4, 14, 17, 18, 23, 24, 25].

In general, existing methods for calculating PnP can be divided based on two criteria. They use either (1) a minimal number of 2D-to-3D correspondences, usually within a RANSAC paradigm to improve robustness, or (2) use more than minimal number of measurement to simplify the equations. Another division can be made between iterative algorithms and non-iterative algorithms, where the former usually requires some approximate solution to begin with.

Previously mentioned methods assume a perspective camera model which is a model physically valid for cameras with a global shutter. However, CMOS sensors that are used in vast majority of today's consumer cameras, smartphones etc. use the rolling shutter (RS) mechanism [16] to capture images. The key difference is that with the global shutter, the entire image is exposed to the light at once, whereas when using the RS the individual image rows (or columns) are captured at different times. When a RS camera moves while capturing the image, several types of distortion such as smear, skew or wobble appear. A perspective camera

model is no longer valid in this case and that can be a problem when using methods that assume this model.

Recent works have shown that RS is an important effect that should be considered in image rectification [10, 19], structure from motion [1, 8, 7] and multiple view stereo [20]. Those works have shown that existing methods can perform poorly on RS data or even fail completely and that incorporating some sort of RS camera model can solve these issues.

In [1] authors tackled the problem of RS absolute pose using a non-linear optimization with the initial guess obtained by a linear method using 8½ points and assuming a planar scene. We present the first minimal solution to the RS absolute pose problem from 6 points only which works for non-planar scenes.

Another non-linear optimization method presented in [7] is suited for video sequences, where camera poses are computed sequentially taking the previous camera pose as an initial guess. In [11], the authors compensated for the RS effect prior to the optimization using estimated camera motion parameters from subsequent video frames. Our solution works for single images exactly as P3P for perspective cameras.

A globally optimal solution using polynomial equations and Gloptipoly [9] solver to solve rolling shutter PnP is shown in [15]. Authors show that the method is capable of providing better results than [1] with the use of seven or more point correspondences. Run time of this approach, which uses Gloptipoly solver, is at least 100 times longer than 1.7ms which we need and therefore [15] it is not a feasible solution for a RANSAC loop. When using more points, this approach is heavily sensitive to mismatches.

1.1. Motivation

Important thing to realize is that the rolling shutter effect is present even in still images, not only video sequences. It is therefore desired to have a R6P method that works on still images that could be used for structure from motion, camera or object localization in cases we don't have a video sequence or are simply limited in computing resources to process every frame of a video. An example could be an unmanned aerial vehicle equipped with a camera for on-board localization or doing a large scale structure from motion reconstruction from a camera mounted on a car.

Contribution. The contribution of this paper is threefold. First, we present the first non-iterative minimal solution to the rolling shutter absolute pose (RnP) problem which improves the precision of camera pose estimates of a standard P3P solution and works on still images as well as video sequences. Second, the feasibility of several different RS camera models is analysed. Third, interesting experimental observations are made showing the benefits and limits of a linearized RS camera model and a standard P3P model on a

rolling shutter data.

We investigate several rolling shutter camera models in section 3, discuss and verify their feasibility for a minimal solution to the RS absolute pose problem. In section 4 we describe how to prepare the equations of the model to be solved by a polynomial solver. Section 5 presents a method how to keep the data close to the linearization point where the selected RS camera model works well. The resulting RS P6P solver is verified and its properties analyzed in detail in section 6.

2. Absolute pose with rolling shutter

The computation of absolute camera pose using 2D and 3D point correspondences (PnP) under the rolling shutter effect brings new challenges. Standard PnP for perspective cameras uses the projection function

$$\lambda_i \mathbf{x}_i = \mathbf{R} \mathbf{X}_i + \mathbf{C} \quad (1)$$

where \mathbf{R} and \mathbf{C} is the rotation and translation bringing a 3D point \mathbf{X}_i from world coordinate system to the camera coordinate system with $\mathbf{x}_i = [r_i, c_i, 1]^\top$, and scalar $\lambda_i \in \mathbb{R}$. When the camera is moving during the image capture, every image row will be captured at different time and hence different positions. \mathbf{R} and \mathbf{C} will therefore be functions of the image row r_i being captured.

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

Next, we will describe functions $\mathbf{R}(r_i)$ and $\mathbf{C}(r_i)$.

3. Rolling shutter camera models

In this section we will consider several rolling shutter camera models and investigate their applicability for the PnP problem.

3.1. SLERP model

To accommodate for camera rotation during frame capture we could interpolate between two orientations. A very popular method how to interpolate rotations is SLERP [21], which was used in [7]. It works with quaternions and the formula to interpolate between two rotations represented by \mathbf{q}_0 and \mathbf{q}_1

$$SLERP(\mathbf{q}_0, \mathbf{q}_1, t) = \mathbf{q}_0 \frac{\sin(\Omega - t\Omega)}{\sin \Omega} - \mathbf{q}_1 \frac{\sin t\Omega}{\sin \Omega} \quad (3)$$

where $\Omega = \arccos(\mathbf{q}_0^\top \mathbf{q}_1)$. It is a linear interpolation on the sphere of quaternions, which in practice means that there will be a constant angular velocity. This is a nice property, which could even hold true in some special cases in reality (e.g. a camera mounted on a rotating platform or a car,

which is turning with constant angular velocity). However, the presence of sine and cosine prevents us from using this model directly in a polynomial solver. We could substitute the sine and cosine with new variables and obtain a polynomial equations but that leads to high order polynomials and too complicated computations to get a fast solver. Moreover, with the increasing number of variables the solution becomes numerically unstable.

3.2. Cayley transform model

Another way how to represent rotations is the Cayley transform [6]. For any vector $\mathbf{a} = [x, y, z]^T \in \mathbb{R}^3$ there is a map

$$R(\mathbf{a}) = \frac{1}{K} \begin{bmatrix} 1+x^2-y^2-z^2 & 2xy-2z & 2y+2xz \\ 2z+2xy & 1-x^2+y^2-z^2 & 2yz-2x \\ 2xz-2y & 2x+2yz & 1-x^2-y^2+z^2 \end{bmatrix} \quad (4)$$

where $K = 1 + x^2 + y^2 + z^2$ which produces the rotation matrix corresponding to the quaternion $w + ix + jy + kz$ normalized so that $w = 1$. The vector \mathbf{a} is a unit vector of the axis of rotation scaled by $\tan \theta/2$. Therefore, 180 degree rotations are prohibited. We can prescribe (2) such that R is a combination using Cayley transform as

$$\lambda_i \begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix} = R(r_i \mathbf{w}) R(\mathbf{v}) \mathbf{X}_i + \mathbf{C} + r_i \mathbf{t} \quad (5)$$

to represent the camera initial orientation by \mathbf{v} and the change of orientation during frame capture by $r_i \mathbf{w}$. This represents a rotation around the axis \mathbf{w} which is close to uniform in angular velocity around $r_i = 0$. Equation (5) is a rational polynomial and we must multiply it by $1+x^2+y^2+z^2$ to get a pure polynomial for the polynomial solver. We obtain a system of polynomial equations of degree five in 18 (3+3+3+3+6 for $\mathbf{C}, \mathbf{v}, \mathbf{t}, \mathbf{w}$ and $\lambda_1 \dots \lambda_6$ respectively) variables which contains 408 monomials. Such a system is difficult to solve and the Gröbner basis solution for this system involves eliminating a 8000x8000 matrix, which is time consuming and numerically not very stable. Due to these reasons we will not consider this model as feasible for our purposes.

3.3. Linearized model

To reduce the degree of the polynomials and the number of monomials we can use a linearization of a rotation matrices. We will linearize around the initial rotation $R(\mathbf{v})$ using the first order Taylor expansion such that

$$\lambda_i \begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix} = (\mathbf{I} + (r_i - r_0)[\mathbf{w}]_x) R(\mathbf{v}) \mathbf{X}_i + \mathbf{C} + (r_i - r_0) \mathbf{t} \quad (6)$$

where $[\mathbf{w}]_x$ is the skew-symmetric matrix

$$[\mathbf{w}]_x = \begin{bmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{bmatrix} \quad (7)$$

Such function model, with the rolling shutter rotation linearized, was used in [15]. This function will deviate from the reality with increasing rolling shutter effect. However, we have observed that this model is usually sufficient for the amount of rolling shutter rotation present in real situations. We now have a system of degree three polynomials in 18 variables with 64 monomials. Unfortunately, we again found this system to be too complicated to be efficiently solved and used in RANSAC based estimation.

3.4. Double linearized model

Let us simplify the model even further by linearizing also the initial rotation. We obtain

$$\lambda_i \begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix} = (\mathbf{I} + (r_i - r_0)[\mathbf{w}]_x)(\mathbf{I} + [\mathbf{v}]_x) \mathbf{X}_i + \mathbf{C} + (r_i - r_0) \mathbf{t} \quad (8)$$

which are simpler polynomial equations of degree two and 28 monomials. The model has an obvious drawback and that is, unlike \mathbf{w} representing the rolling shutter motion and being presumably small, \mathbf{v} can be arbitrary. Therefore, the model's accuracy would depend on the initial orientation of the camera in the world frame. A possible solution is to force \mathbf{v} to be close to zero and we propose a way how to do this in section 5. In section 4 we will show how to modify (8) to make the computation more efficient.

4. R6P for RANSAC

In previous section we presented several rolling shutter camera models and in this section we show how to use the double linearized model in a RANSAC environment.

4.1. Solving the equations

To solve the polynomial equations of the double linearized model (8), we used the Gröbner basis method [3]. This method for solving systems of polynomial equations has been recently used to create very fast, efficient and numerically stable solvers to many difficult problems. The method is based on polynomial ideal theory and special bases of the ideals called Gröbner bases [3]. To create an efficient Gröbner basis solver for the absolute pose rolling shutter problem we used the automatic generator of Gröbner basis solvers [13].

The minimal number of 2D-to-3D point correspondences necessary to solve the absolute pose rolling shutter problem is six. For six point correspondence, the double linearized model (8) results in quite a complex system of

$3 \times 6 = 18$ equations in 18 unknowns. Such a system is not easy to solve for the Gröbner basis method and therefore it has to be simplified.

To simplify the input system (8) we first eliminate the scalar values λ_i by multiplying all equations (8) from the left by the skew symmetric matrix

$$\begin{bmatrix} 0 & -1 & c_i \\ 1 & 0 & -r_i \\ -c_i & r_i & 0 \end{bmatrix}. \quad (9)$$

This leads to 18 equations, from which only 12 are linearly independent, in 22 monomials and 12 unknowns, i.e., the rotation parameters w, v and the translation parameters C and t .

The 12 linearly independent equations are linear in the unknown translation parameters C and t . Therefore, the translation parameters C and t can be easily eliminated from these equations. This can be done either by performing Gauss-Jordan (G-J) elimination of a matrix representing the 12 linearly independent equations or by expressing the six translation parameters as functions of the rotation parameters w and v and substituting these expressions to the remaining six equations.

After the simplification we obtain a final system of six equations in six unknowns and 16 monomials. This system has 20 solutions. The Gröbner basis solver of the proposed R6P rolling shutter problem starts with these six equations in six unknowns, i.e., the rotation parameters w, v .

From the automatic generator [13], we obtained an elimination template which encodes how to multiply these six input polynomial equations by the monomials and then how to eliminate the polynomials using G-J elimination to obtain all polynomials necessary for solving the problem.

To get the elimination template, the automatic generator first generated all monomial multiples of the initial six equations up to the total degree of five. This resulted in 504 polynomials in 462 monomials. Then the generator removed all unnecessary polynomials and monomials. This resulted in a final 196×216 elimination template matrix. Note that this elimination template matrix needs to be found only once in the preprocessing phase.

The final solver for the P6P rolling shutter problem then only performs one G-J elimination of the 196×216 elimination template matrix. This matrix contains coefficients which arise from specific measurements, i.e., six 2D-to-3D point correspondences. Then the solutions to the rotation parameters are found from the eigenvectors of a special 20×20 multiplication matrix created from the rows of the eliminated template matrix. This gives us a set of up to 20 real solutions to w and v . By substituting these solutions to the equations (8) we obtain solutions to C and t .

Usually only one of these 20 solutions is geometrically feasible, i.e., is real and of a reasonable values of param-

eters. Specifically, if we consider only reasonable values of the rolling shutter angular velocity w we can eliminate many solutions that are not feasible. Authors of [15] used the same linearization for w and showed that when $\|w\| > 0.05$ the model loses its accuracy. We decided to discard solutions with $\|w\| > 0.2$ which corresponds to angular velocity of approximately 11deg/s. Solutions beyond this threshold are not interesting, since they are far from the linearization point. In our experiments, this criterion successfully eliminated 90-95% of solutions to be verified by RANSAC, which speeded up the process significantly by avoiding lots of work in model verification.

The final R6P solver takes about 1.7 ms in MATLAB on a 3.2 GHz i7 CPU. An improvement can be expected using C++. Moreover, the 196×216 elimination template matrix used in this R6P Gröbner basis solver is quite sparse and therefore the final solver can be even speeded up using the recently published SBB method for speeding-up Gröbner basis solvers [12]. The expected speed up of the G-J elimination part of the solver, which now takes 0.9ms, is $3 \times 5 \times$.

5. Getting close to the linearization point

As mentioned in section 3, the double linearized model will only be a good approximation when close to the linearization point. That is the case when R is close to I . We can enforce this condition if we have an approximation R_a to R . Then we can transform the 3D points as

$$\hat{X}_i = R_a X_i \quad i = 1, \dots, 6 \quad (10)$$

and replace X_i in (8) by \hat{X}_i . Such solution R_l should then be close to I and we can obtain R as $R_l R_a$. To get such approximation we can use for example an inertial sensor which is often present in cellphones, cameras or on-board a robot or UAV. However, we don't want to limit ourselves to having additional sensor information so we propose to obtain R_a using a standard P3P algorithm [5]. We will show in the experiments to which limits this approach works and that it can indeed provide a sufficient approximation for our solver to work well.

Notice that this approach requires only an approximation to camera orientation not the camera position.

6. Experiments

We conducted several experiments on synthetic as well as real datasets. The synthetic experiments were aimed at analyzing the properties of the double linearized rolling shutter camera model, which brings an interesting insight on how the solver will behave under different conditions. On the real datasets, in the absence of ground truth, we focused on the number of matches classified as inliers using RANSAC. This corresponds to a typical application of absolute pose algorithm, where we want our model to be able to fit as

many matches as possible while avoiding the mismatches. We compared our R6P only to P3P [4], since to the best of our knowledge it is the only alternative with the same applicability. It is important to note that [7] can be used only on video sequences, [1] requires 9 co-planar points and [15] uses global optimization which is sensitive to mismatches and due to speed of Gloptipoly too slow for the RANSAC paradigm.

In the synthetic datasets, a calibrated camera was considered with field of view of 45 degrees. It was randomly placed in a distance of $\langle 1; 3.3 \rangle$ from the origin, observing a group of 3D points randomly placed in a cube with side length 2 centered around the origin. Camera initial orientation was different based on the type of experiment. The rolling shutter movement was simulated using Cayley parameterization, because using the double linearized model for generating data would allow the solver to always find the exact solution. The image projections were obtained by solving (5) for r and c . Six points were then randomly chosen from the projections to solve for the camera parameters. Since the solver can return up to 20 real solutions, the one closest to the ground truth was always chosen, as it would be probably chosen in the RANSAC estimation.

The first two experiments focus on varying either of the two rolling shutter parameters, i.e. the translational and angular velocity. The camera orientation is kept $R = I$ so we avoid the effect of the initial camera orientation linearization. We varied the angular velocity approximately from 0 to 30 degrees per frame. Angular velocity 30 deg/frame means that the camera moved by 30 degrees between acquiring the first and the last row. The translational velocity was varied from 0 to 1 which is approximately 50% of the average distance of the camera center from the 3D points. The results are shown in Figure 2. As expected, because the model does not exactly fit the data (as will be the case in real data), with increasing rolling shutter effect the performance of the solver decreases. However, the results are very promising since even at higher angular velocities the solver still delivers fairly precise results. At 30 deg/frame the mean orientation error is still below half a degree and the position error is less than half a percent. When varying the translation velocity only, we found the solver to be giving exact results up to the numerical precision, which was expected, since the model fits exactly the data.

In the third experiment we varied both rolling shutter parameters, i.e. translational and angular velocity to see the combined effect. Again, the camera orientation is kept $R = I$ and the same ranges are used for parameter variation as in first two experiments. The results, see Figure 3, are similar to those in the first experiment, indicating that the presence of a constant velocity translational rolling shutter effect does not affect the results of the solver significantly.

The fourth experiment was focused on finding out how

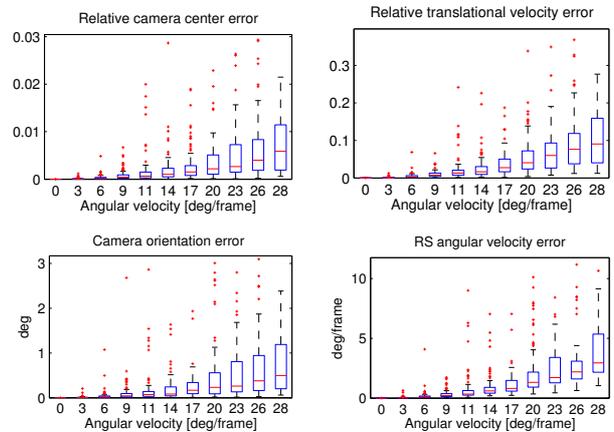


Figure 2: Results for varying RS angular velocity without RS translation.

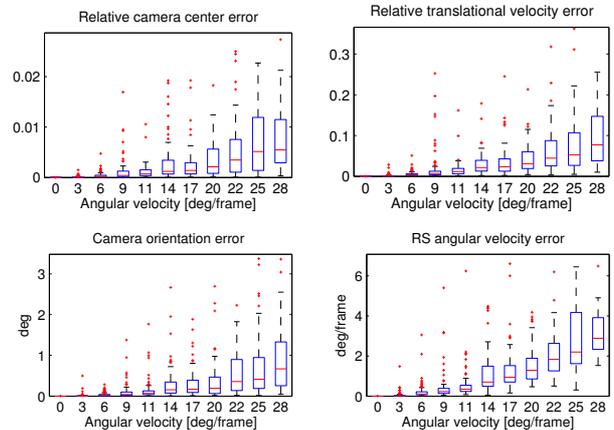


Figure 3: Results for varying both RS angular velocity and RS translation.

well the model behaves around its linearization point, i.e. when $R \neq I$. Rolling shutter parameters were uniformly chosen from values $\langle 0; 20 \rangle$ deg/frame for the angular velocity and $\langle 0; 0.2 \rangle$ for the translational velocity, which is approximately ten percent of the average distance of camera center from the 3D points. Camera orientation was varied in the interval of $\langle 0; 33 \rangle$ degrees. Results are in figure 4 and they show that the model is very prone to error when being far from its linearization point, with the mean camera orientation error going up to five degrees and mean relative camera center error approaching 0.4 when the camera is rotated 33 degrees away from the linearization point. C and R are computed quite accurately, when R is within approx. 10 degrees from I . It suggests that if we can use some standard non-RS method, such as P3P to find an initial R_0 to align the data, we can then apply our solver to get a more accurate camera pose. We tested this approach in experiment 7.

In synthetic experiments 5 and 6, we compared our R6P solver with a standard P3P solver [5] under the same conditions as in experiments 3 and 4, i.e. varying both rolling shutter parameters and varying camera orientation respec-

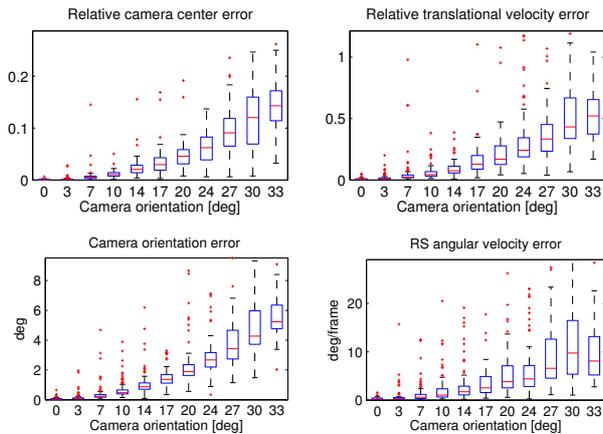


Figure 4: Experiment 4 - varying camera orientation, forcing the model away from its linearization point.

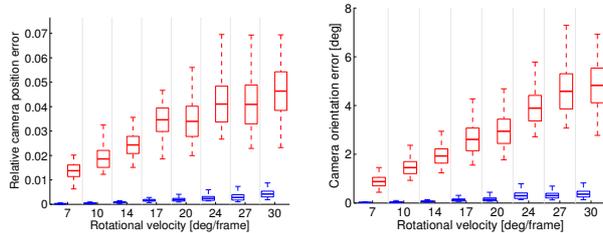


Figure 5: Experiment 5 - varying RS parameters (angular and translational velocity) and comparing our R6P solver (blue) against standard P3P solver (red). The camera orientation is $R = I$ and thus at the linearization point of the model.

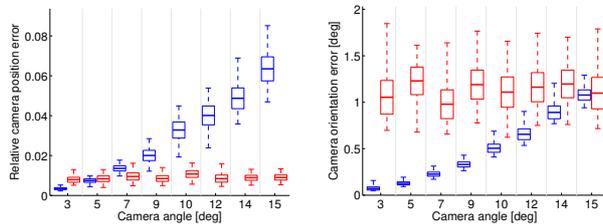


Figure 6: Experiment 6 - randomly choosing RS parameters and increasing the camera orientation angle. Same conditions as in experiment 4, but here comparing our R6P solver (blue) with standard P3P solver (red).

tively. The results in figure 5 show that at the linearization point, our R6P solver is far superior to the standard P3P method in estimating both camera position and orientation. Results in figure 6 hint that the linearization is the key issue in our R6P solver and that around 7 degrees of distance from $R = I$ our solver is surpassed in precision of estimating the camera center and at 14 to 17 degrees in the precision of estimating camera orientation. Note the interesting discrepancy between the error in camera position and orientation. An interesting thing to notice from these two experiments is that the global P3P solver is capable of bringing the camera orientation within six degrees of the ground truth, even under large RS effect. At that point, if we apply our solver, the precision should improve significantly to values below 0.5 deg. To verify the previous is the

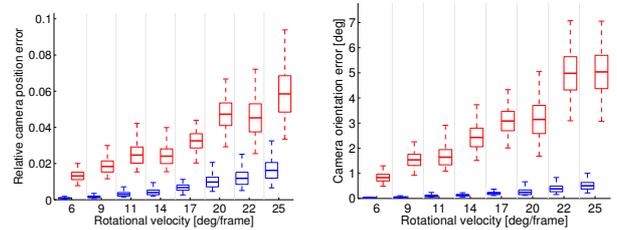


Figure 7: Experiment 7 - random camera orientation and increasing RS angular and translational velocity. Results after applying P3P (red) and subsequently applying R6P (blue). Significant improvement for both camera position and orientation is made using P6P after P3P.

purpose of experiment 7. The camera orientation was chosen randomly and the RS parameters were increasing as in experiments 3 and 5. The results confirm our hypothesis. If the global P3P, or any other method, is able to compute the camera orientation within 15 degree error then our solver improves the solution to an average error below one degree. Interesting observation is that unlike in experiment 6 here the precision is significantly better for both camera center and orientation.

6.1. Real data

For real experiments we used the four datasets from [8] where an Iphone 4 camera was placed together with a global shutter Canon camera on a rig. We acquired two other datasets using similar camera rig, but with a high-speed global-shutter camera and a Nexus 4 cellphone. Videos were taken when moving this rig by hand or walking around. We therefore have for each dataset 2 sets of images of the same scene. One set is with rolling shutter effect and one with global shutter.

6.2. Obtaining 2D-3D correspondences

To see the behavior of our method on real data, we needed to obtain 3D to 2D correspondences for the rolling shutter images. We decided to do a reconstruction using a standard SfM pipeline [22] using the global shutter images first. Then, we matched the rolling shutter images with the global shutter matches that had a corresponding 3D point in the global shutter 3D model. That way we obtained the correspondences between 2D rolling shutter features and 3D global shutter points. It was verified visually that this approach provided 2D-3D correspondences with a very small number of mismatches, i.e. a 2D correspondence being matched to a wrong 3D point. This is probably due to the fact, that all 3D points have already gone through an SfM pipeline and only good 3D points which were successfully matched in several cameras remained. Still, some mismatches were present, but according to our experiments, this number was not higher than 10%.



Figure 8: Example images of all datasets used.

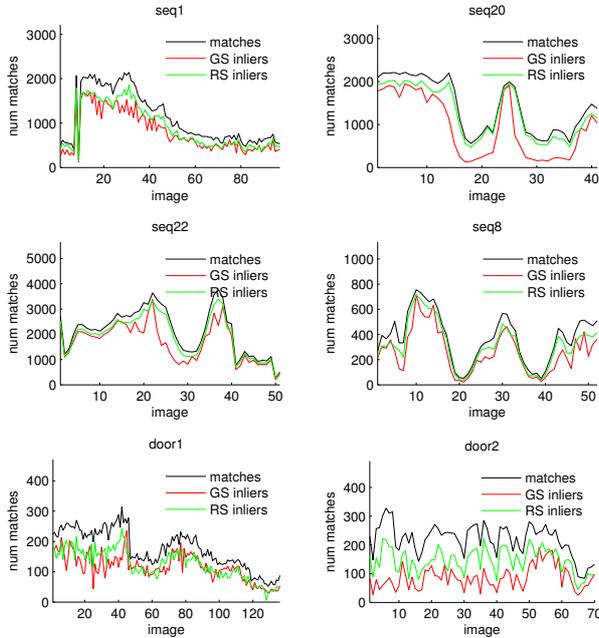


Figure 9: Experiments on real data. Number of inliers after running 100 rounds of RANSAC. Number of 2D-3D matches from global shutter images to rolling shutter images are in black, number of inliers obtained by P3P are in red and number of inliers obtained by R6P are in green. The results are averaged over 100 runs to reduce randomness.

6.3. Evaluation

To evaluate our method, we measured the number of inliers, i.e. the 2D-3D correspondences in agreement with the model, after performing RANSAC. This is an important measure, since a common use of PnP is to calculate the camera pose and tentative 3D points for triangulation. The more points will be classified as inliers the more points will appear in the reconstruction and will support further cameras.

We applied first the global P3P, transforming the 3D points and then using our R6P solver as described in section 5. Since our data contained only few mismatches, 100 iterations of RANSAC proved to be enough to obtain a good camera pose. To reduce randomness of RANSAC results, we averaged the numbers over 100 successive RANSAC runs. As it is seen in figure 9, R6P is able to classify more points as inliers compared to P3P. The difference is significant especially when camera moves rapidly and/or the scene is close to the camera. This result confirms our expectation that as the camera movement during the capture becomes larger the need for a rolling shutter model is more significant. Datasets seq20, seq8 and door2 contained more cam-

era motion and therefore show a larger gap between results of P3P and R6P.

A good example is in dataset seq20, where the camera is fairly still in the beginning, then undergoes a rapid change in orientation (going upwards following the trunk of a palm tree), stops and then goes down again. The number of inliers returned by R6P and P3P when the camera is still is comparable, although higher for R6P since there are some RS distortions caused by handshake. As soon as the camera starts moving, the number of inliers for P3P drops drastically, sometimes even below 10% of the number of matches. R6P, in contrast, manages to keep the number of inliers above 90% of the number of matches. That is a huge difference.

Important observation is, that even though P3P fails to classify more than 80% of the matches as inliers it still provides a sufficient estimate of the camera orientation for the R6P to produce much better result. A detailed visualization of one of the results on seq20 is given in figure 10.

7. Conclusion

In this paper, we addressed the problem of perspective-n-point for cameras with rolling shutter. We presented four different models which capture camera translation and rotation during frame capture. Only one of them was found to be feasible for using a polynomial solver. Using this model, camera position, orientation, translational velocity and angular velocity can be computed using six 2D-to-3D correspondences. This model uses a linear approximation to the camera orientation and therefore is only meaningful around unitary camera orientation matrix. We analyzed its behavior around the linearization point and concluded that it is only reasonable to use it within approximately ten degree region around the linearization point. We also showed that standard P3P algorithm is able to estimate the camera orientation with an error under six degrees for camera movements as high as 30 degrees over image capture time combined with a translational movement up to 10% of the camera distance from the object. Using these findings we proposed to first apply a standard P3P algorithm, transform the 3D points using the estimated rotation to bring the estimated camera pose close to unitary matrix and then apply our R6P solver. This approach turned out to work very well delivering average camera orientation error under half a degree and relative camera position error under 2% even for large rolling shutter distortions. It was verified to work on real data, delivering increased number of inliers when using R6P over P3P in RANSAC. These results suggest that having an initial guess on camera orientation, such as from an inertial measurement unit present in cellphones or UAV's, one could use our R6P solver directly.

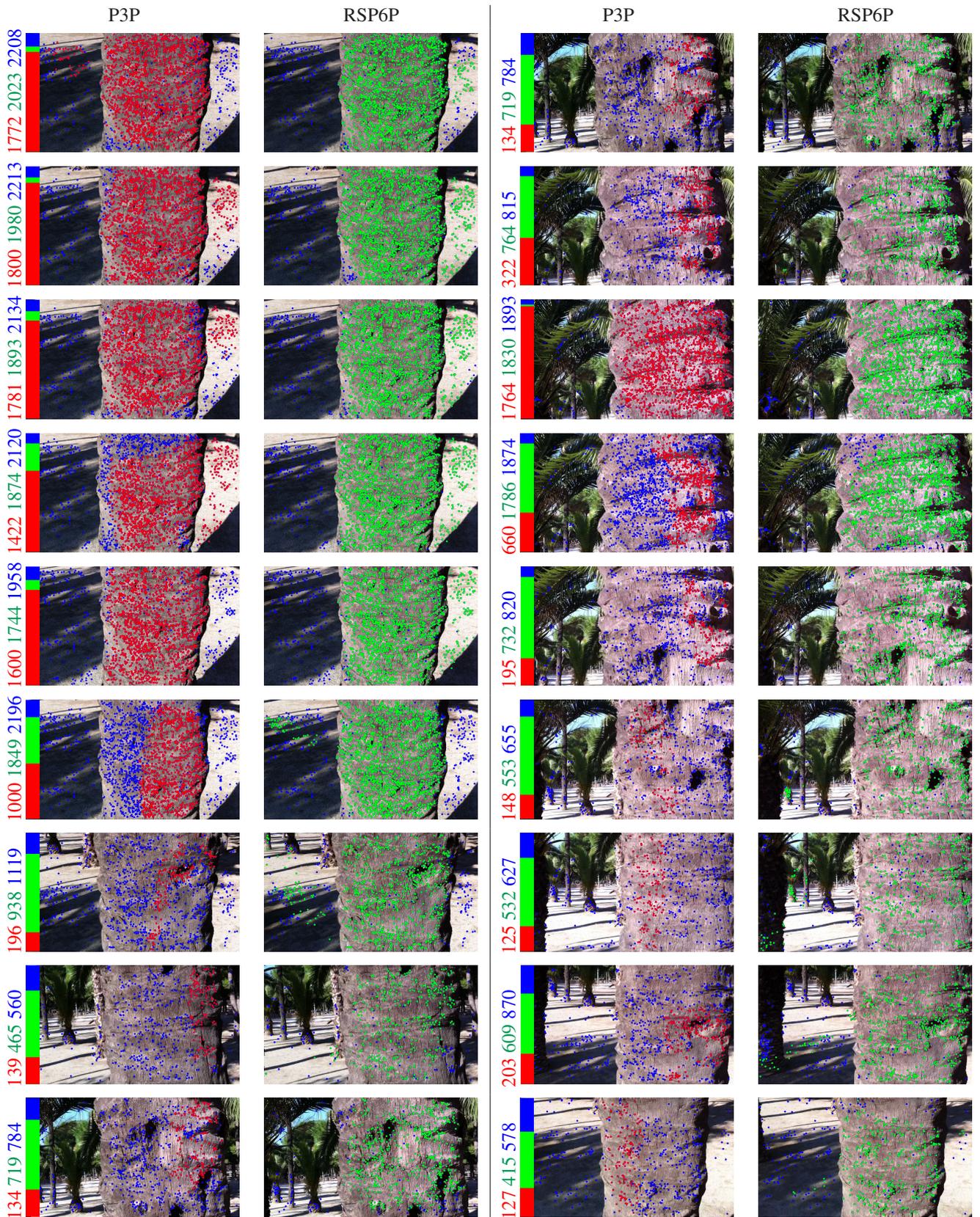


Figure 10: Results on dataset seq20, matched correspondences are in blue, inliers after RANSAC using P3P and R6P are in red and green respectively. The actual numbers of inliers are displayed on side of each image pair.

Acknowledgment

This research was supported by EC project FP7-SPACE-2012-312377 PRoViDE and Grant Agency of the CTU Prague project SGS13/202/OHK3/3T/13.

References

- [1] O. Ait-aider, N. Andreff, J. M. Lavest, U. Blaise, P. C. Ferr, and L. U. Cnrs. Simultaneous object pose and velocity computation using a single view from a rolling shutter camera. In *In Proc. European Conference on Computer Vision*, pages 56–68, 2006.
- [2] M. André Ameller, B. Triggs, and L. Quan. Camera pose revisited: New linear algorithms. In *14eme Congres Franco-phonie de Reconnaissance des Formes et Intelligence Artificielle. Paper in French*, page 2002, 2002.
- [3] D. Cox, J. Little, and D. O’Shea. *Using Algebraic Geometry*. Graduate Texts in Mathematics. Springer, 2005.
- [4] 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(6):381–395, June 1981.
- [5] R. Haralick, D. Lee, K. Ottenburg, and M. Nolle. Analysis and solutions of the three point perspective pose estimation problem. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR ’91., IEEE Computer Society Conference on*, pages 592–598, Jun 1991.
- [6] M. Hazewinkel. *Encyclopaedia of mathematics*. Springer-Verlag, Berlin New York, 2002.
- [7] J. Hedborg, P.-E. Forssén, M. Felsberg, and E. Ringaby. Rolling shutter bundle adjustment. In *CVPR*, pages 1434–1441, 2012.
- [8] J. Hedborg, E. Ringaby, P.-E. Forssen, and M. Felsberg. Structure and motion estimation from rolling shutter video. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 17–23, 2011.
- [9] D. Henrion, J.-B. Lasserre, and J. Lofberg. Gloptipoly 3: Moments, optimization and semidefinite programming. *Optimization Methods Software*, 24(4-5):761–779, Aug. 2009.
- [10] C. Jia and B. L. Evans. Probabilistic 3-d motion estimation for rolling shutter video rectification from visual and inertial measurements. In *MMSP*, pages 203–208. IEEE, 2012.
- [11] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality, ISMAR ’09*, pages 83–86, Washington, DC, USA, 2009. IEEE Computer Society.
- [12] Z. Kukulova, M. Bujnak, J.Heller, and T. Pajdla. Singly-bordered block-diagonal form for minimal problem solvers. In *ACCV’14*, 2014.
- [13] Z. Kukulova, M. Bujnak, and T. Pajdla. Automatic generator of minimal problem solvers. In D. A. Forsyth, P. H. S. Torr, and A. Zisserman, editors, *Computer Vision - ECCV 2008, 10th European Conference on Computer Vision, Proceedings, Part III*, volume 5304 of *Lecture Notes in Computer Science*, pages 302–315, Berlin, Germany, October 2008. Springer.
- [14] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81(2):155–166, 2009.
- [15] L. Magerand, A. Bartoli, O. Ait-Aider, and D. Pizarro. Global optimization of object pose and motion from a single rolling shutter image with automatic 2d-3d matching. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part I, ECCV’12*, pages 456–469, Berlin, Heidelberg, 2012. Springer-Verlag.
- [16] M. Meingast, C. Geyer, and S. Sastry. Geometric Models of Rolling-Shutter Cameras. *Computing Research Repository*, abs/cs/050, 2005.
- [17] L. Quan and Z. Lan. Linear n-point camera pose determination. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(8):774–780, Aug 1999.
- [18] G. Reid, J. Tang, and L. Zhi. A complete symbolic-numeric linear method for camera pose determination. In *Proceedings of the 2003 International Symposium on Symbolic and Algebraic Computation, ISSAC ’03*, pages 215–223, New York, NY, USA, 2003. ACM.
- [19] E. Ringaby and P.-E. Forssén. Efficient video rectification and stabilisation for cell-phones. *International Journal of Computer Vision*, 96(3):335–352, 2012.
- [20] O. Saurer, K. Koser, J.-Y. Bouguet, and M. Pollefeys. Rolling shutter stereo. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 465–472, Dec 2013.
- [21] K. Shoemake. Animating rotation with quaternion curves. *SIGGRAPH Comput. Graph.*, 19(3):245–254, July 1985.
- [22] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *Int. J. Comput. Vision*, 80(2):189–210, Nov. 2008.
- [23] B. Triggs. Camera pose and calibration from 4 or 5 known 3d points. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 278–284 vol.1, 1999.
- [24] Y. Wu and Z. Hu. Pnp problem revisited. *Journal of Mathematical Imaging and Vision*, 24(1):131–141, 2006.
- [25] L. Zhi and J. Tang. A complete linear 4-point algorithm for camera pose determination, 2002.