# 3D reconstruction with time variant geometry

A Dissertation Presented to the Faculty of the Electrical Engineering of the Czech Technical University in Prague in Partial Fulfillment of the Requirements for the Ph.D. Degree in Study Programme No. P2612 - Electrotechnics and Informatics, branch No. 3902V035 - Artificial Intelligence and Biocybernetics, by


**Čeněk Albl**


Prague, August 2018

Thesis Advisor

**doc. Ing. Tomáš Pajdla, Ph.D.**

Thesis Co-advisor

**RNDr. Zuzana Kúkelová, Ph.D.**


Center for Machine Perception
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
Karlovo náměstí 13, 121 35 Prague 2, Czech Republic
phone: +420 224 357 666
http://cmp.felk.cvut.cz

Applied Algebra and Geometry
Czech Institute of Informatics, Robotics and Cybernetics
Czech Technical University in Prague
Jugoslávských Partyzánů, 160 00 Prague 6, Czech Republic
phone: +420 224 354 139
http://ciirc.cvut.cz

# Abstract

Methods for reconstructing 3D models of our environment have been long established and well studied. The majority of 3D computer vision techniques rely on the perspective camera model, which does not change its parameters over time. However, there are many cases where this assumption does not accurately fit the reality of the image capture process.

For example, a majority of today's cameras use CMOS sensors with a rolling shutter, which exposes every image line at a different time. When such a camera is moving, e.g., in hand-held videos, aerial footage, or augmented reality applications, the images will contain distortions. These distortions cause standard computer vision algorithms to fail. We need a different camera model that can describe the camera motion as well in order to make the standard methods work to our satisfaction.

This work proposes new rolling-shutter adapted algorithms and presents models that describe camera motion during the image capture and, therefore, provide improved results. Several algorithms for computing rolling shutter camera absolute pose are proposed and verified on real data and show improved results over the state-of-the-art methods. A version of the camera absolute pose algorithm that utilizes the inertial measurement unit, in particular, the gravity vector is also presented. Simple linearized iterative versions of the rolling shutter absolute pose solvers are derived, providing comparable performance and superior speed.

All presented solutions require only a minimal number of points and are fast, which makes them ideal candidates for robust estimation using the RANSAC paradigm. Their fast run-time makes them suitable for real-time applications such as augmented reality.

Bundle adjustment is a key technique in 3D computer vision that jointly optimizes the camera parameters and 3D scene parameters. We discuss the problem of a general bundle adjustment method with rolling shutter camera models and point out an important degeneracy that causes the optimization to inevitably choose a wrong local minimum. Due to the nature of the degenerate configuration, it is easy to avoid it if the images are taken in a specific way, which we verify on synthetic as well as real data.

Another case where standard algorithms that do not consider time variant geometry can fail is an unsynchronized multi-camera system. Multi-camera systems have many applications from autonomous driving to human motion capture. Various systems are being developed and need to be calibrated and time synchronized in order to work properly. Calibrating multi-camera systems can be cumbersome and time-consuming especially when cameras are not hardware synchronized, which is often difficult or impossible. Calibration from moving objects in the scene is desired, but to obtain correct image correspondences, one needs precise time synchronization.

We present a solution to jointly estimate two-camera geometry, i.e. the fundamental or homography matrix, and time shift, enabling the calibration of two cameras in time and in space from motion in image sequences. The solution is again based on a minimal number of points and is efficient enough to be used in RANSAC. The method does not require any special image

content, apart from the scene not being completely static. Further, an iterative algorithm on top of the direct solution is proposed, allowing the synchronization of sequences with large time offsets on the order of seconds, which was verified on publicly available real datasets.

# Abstrakt

Metody rekonstrukce 3D modelů okolního světa jsou již dlouho studovaným a důkladně probádaným tématem. Většina technik 3D počítačového vidění je postavená na perspektivním modelu kamery, jehož parametry se nemění v čase. Existuje ovšem spousta případů, kdy tento předpoklad neplatí.

Například, většina dnešních fotoaparátů používá CMOS sensor s elektronickou plovoucí uzávěrkou, který exponuje každý řádek obrazu v jiný čas. Pokud se takováto kamera pohybuje, jako je tomu například u natáčení videa rukou, u leteckého snímkování, nebo u rozšířené reality, výsledný obraz bude obsahovat zkreslení. Tato zkreslení mají za následek nefunkčnost standardních algoritmů počítačového vidění. Abychom v takovém případě dosáhli uspokojivých výsledků, potřebujeme standardní metody rozšířit o nový model kamery.

Tato práce navrhuje nové algoritmy upravené pro práci s plovoucí uzávěrkou a představuje nové modely kamery, které popisují její pohyb během snímání a díky tomu poskytují lepší výsledky ve výše popsaných situacích. Několik algoritmů pro výpočet absolutní polohy kamery je navrženo a otestováno jak na syntetických datech, tak na reálných scénách, kde vykazují lepší výsledky než předchozí metody. Verze algoritmu pro absolutní polohu kamery s využitím gravitačního vektoru získaného z inerciální měřící jednotky je také navržena a otestována. Jednoduché, lineární, iterativní algoritmy pro výpočet absolutní polohy kamery s plovoucí uzávěrkou jsou odvozeny, poskytujíc porovnatelné výsledky, ale větší rychlost.

Všechna navrhovaná řešení potřebují pouze minimální počet korespondencí mezi scénou a obrazem a jsou velmi rychlé, což z nich dělá vhodné kandidáty pro robustní odhad parametrů pomocí schématu RANSAC. Jejich rychlost je také činí vhodnými pro aplikace v reálném čase, jako je například rozšířená realita.

Metoda vyrovnání svazku paprsků je klíčovou technikou 3D počítačového vidění, která optimalizuje zároveň parametry kamer i 3D scény. V této práci diskutujeme problém nasazení nových modelů popisujících pohyb kamery v metodě vyrovnání svazku paprsků a poukazujeme na kritické konfigurace, které vedou optimalizační metodu k nevyhnutelnému pádu do špatného lokálního minima. Z principu této kritické konfigurace je zřejmé, že se jí lze v praxi vyhnout pořizováním fotografií specifickým způsobem, což jsme ověřili na syntetických i reálných datech.

Dalším případem, kdy standardní metody, které neuvažují časově proměnlivou geometrii kamer a scény, selhávají, je nesynchronizovaný systém více kamer. Multi-kamerové systémy mají mnoho využití od autonomního řízení automobilů po snímání lidského pohybu. Různé systémy obsahující více kamer jsou vyvíjeny a je nutná jejich kalibrace a synchronizace v čase, aby mohly fungovat správně. Kalibrace multi-kamerových systémů může být velmi obtížná a časově náročná, obzvlášť pokud nejsou hardwarově časově synchronizované, čehož je většinou obtížné až nemožné dosáhnout. Automatická kalibrace z objektů pohybujících se v prostoru je

vhodná, ale k získání přesných obrazových korespondencí je zapotřebí přesné časové synchronizace.

V této práci představujeme řešení problému současného odhadu vzájemné polohy dvou kamer, tedy fundamentální matice nebo homografie, spolu s odhadem časového posunu, což umožňuje kalibraci dvou a více kamer v prostoru a čase s využitím pohybu v obraze. K řešení je opět použit minimální počet bodů a výsledné algoritmy jsou dostatečně rychlé pro robustní odhad pomocí RANSAC. Naše metoda nevyžaduje žádný specifický obrazový obsah, kromě dostatečně dlouhého pohybu. Dále popisujeme iterativní algoritmus postavený na navržených minimálních algoritmech, který dovoluje časovou a prostorovou synchronizaci obrazových sekvencí s velkým časovým posunem, v řádech sekund, což bylo ověřeno na veřejně dostupných reálných datech.

# Authorship

I hereby certify that the results presented in this thesis were achieved during my own research in cooperation with my thesis advisor doc. Ing. Tomáš Pajdla, Ph.D. and my co-advisor RNDr. Zuzana Kúkelová, Ph.D.

# Acknowledgements

I would like to express my thanks to my colleagues and friends at CMP and later at CIIRC with whom I had the pleasure of working with. I am grateful to my supervisor, Tomáš Pajdla, for his guidance, brilliant ideas, endless patience and a very practical approach to solving problems and handling deadline rushes. Special thanks go to my co-advisor and friend Zuzana Kúkelová who introduced me to the topic of polynomial solvers, and was my closest collaborator on the topics covered by this thesis, always providing me with interesting problems to solve.

I want to thank prof. Akihiro Sugimoto and Andrew W. Fitzgibbon with whom I had the pleasure of working with as an intern. It was a great experience and I am glad for the cooperation we maintained afterward.

Last, but definitely not least, I am very grateful to my family, friends and especially to my loving wife who always supported me on this difficult and long journey.

I gratefully acknowledge the support of EC H2020-ICT-731970 project LADIO and project IMPACT CZ.02.1.01/0.0/0.0/15_003/0000468 which is funded by EU Structural and Investment Funds, Operational Programe Research, Development and Education.. Partial support of the Grant Agency of the Czech Technical University under project SGS13/202/OHK3/3T/13, and Grant-in-Aid for Scientific Research of the Ministry of Education, Culture, Sports, Science and Technology of Japan is also acknowledged.

# Contents

**Keywords:** Rolling shutter, Minimal problems, Polynomial solvers, Structure from Motion, Bundle Adjustment, Absolute Pose, Relative Pose, Camera synchronization

# Notation

| | |
|---|---|
| $h$ | scalar value |
| $\mathbf{h}$ | column vector |
| $\mathtt{H}$ | matrix |
| $\mathcal{H}$ | set, list |
| $H$ | entity (image, camera, descriptor, model, ...) |
| $h_i$ | scalar $h$ with label/index $i$ |
| $\mathbf{h}_i$ | vector $\mathbf{h}$ with label/index $i$ |
| $\mathtt{H}_i$ | matrix $\mathtt{H}$ with label/index $i$ |
| $\mathcal{H}_i$ | set/list $\mathcal{H}$ with label/index $i$ |
| $H_i$ | entity $H$ with label/index $i$ |
| $h(.)$ | functions which map domain "." to scalars |
| $\mathbf{h}(.)$ | functions which map domain "." to vectors |
| $\mathtt{H}(i,j)$ | $[i,j]$-th element of matrix $\mathtt{H}$ |
| $h \in \mathcal{H}$ | means that $h$ is element of set/list $\mathcal{H}$ |
| $\mathbf{0} = [0,\ldots,0]^\top$ | vector of zeros |
| $\mathtt{I} = \mathrm{diag}([1,\ldots,1]^\top)$ | identity matrix |
| $\lvert\mathcal{H}\rvert$ | number of elements of set/list $\mathcal{H}$ |
| $\lVert\mathbf{h}\rVert$ | Euclidean ($L_2$) norm of vector $\mathbf{h}$ |
| $[\mathbf{h}]_\times = \begin{bmatrix} 0 & -h_3 & h_2 \\ h_3 & 0 & -h_1 \\ -h_2 & h_1 & 0 \end{bmatrix}$ | cross product matrix of vector $\mathbf{h}$ |

# Commonly used symbols and abbreviations

| | |
|---|---|
| RS | Rolling Shutter |
| GS | Global Shutter |
| SfM | Structure from Motion |
| BA | Bundle Adjustment |
| VSLAM | Visual Simultaneous Localization And Mapping |
| AR | Augmented Reality |
| IMU | Inertial Measurement Unit |
| GB | Gröbner Basis |
| RANSAC | Random Sample Consensus |
| EG | Epipolar Geometry |
| PnP | Perspective-n-Point |
| RnP | Rolling-shutter-n-Point |
| $I$ | image |
| $\mathbf{X}$ | 3D point |
| $\mathbf{u}$ | image point (projection of a 3D point) |
| $\mathtt{R}$ | camera orientation |
| $\mathbf{T}$ | camera translation (in camera reference frame) |
| $\mathbf{C}$ | camera translation (in world reference frame) |
| $\mathtt{K}$ | camera calibration matrix (internal parameters) |
| $\mathbf{t}$ | camera translational velocity |
| $\boldsymbol{\omega}$ | camera rotational velocity |
| $r$ | image row |
| $c$ | image column |
| $\mathtt{P} = \mathtt{K}[\mathtt{R} \mid \mathbf{T}]$ | camera projection matrix |
| $\mathtt{F}$ | fundamental matrix |

# 1 Introduction

3D computer vision encompasses many methods that in general try to use images or depth maps to create a 3-dimensional representation of the surrounding. Structure from Motion (SfM) attempts to reconstruct the scene from multiple images while also computing the poses of the cameras that captured these images. These poses can be further used in Multi-View Stereo that provides visually appealing dense 3D models. A similar method as SfM, called Visual Simultaneous Localization and Mapping is used in the robotics field, building a map of the environment and localizing the robot. Augmented reality computes the camera pose of the current viewpoint in order to place virtual objects in the scene.

In all these examples we use models that describe how an image is created from the light rays reflecting off the 3D objects. These models are an approximation of the actual physical process that simplifies it and allows us to efficiently compute the particular task. We will call these models of cameras and 3D objects as the scene and camera geometry.

In this thesis, we deal with cases when the scene and (or) the camera geometry changes over time which frequently happens in practice. Standard computer vision techniques that don't take the changing geometry into account fail to produce satisfactory results. In particular, one of the frequent cases is when images are captured by a Rolling Shutter camera that moves or captures a moving object. The distortions produced in the image cannot be described by standard, time-independent geometry. The second case is when a de-synchronization between two image sequences occurs and the standard two-view geometry no longer applies.

We present methods that solve several classical computer vision problems for Rolling Shutter cameras and unsynchronized multi-camera setups. These methods allow providing better results across a wide variety of computer vision applications.

## 1.1 3D with Rolling shutter

Rolling shutter is a technique of capturing images that has overtaken the domain of consumer cameras today. Almost all cellphones, action cameras, mirror as well as mirror-less cameras even from the professional market part of the spectrum are equipped with CMOS sensors using a rolling shutter. Moreover, due to the low price of such sensors, industrial manufacturers often incorporate rolling shutter cameras even in fields where high precision is required such as automotive and robotics.

The other dominant shutter type is the global shutter. The difference between these two is that with global shutter the entire image is exposed to the light at once, whereas rolling shutter

Figure 1.1: Examples of rolling shutter induced image distortions. A fast moving object in a still shot (left), courtesy of Soren Ragsdale. Translational motion of the camera (center) and rotational motion of the camera (right), courtesy of Greg Kopec. In this thesis we will deal with the cases such as the two examples from the right.

exposes the image line by line, with each row or column (depending on the sensor) being exposed at a different time, see figure 1.2.

As long as the camera is not moving and the scene is completely static during the image capture, there will be no difference between an image captured using the global or rolling shutter. If, however, the camera or the object being captured moves, rolling shutter camera will produce a distorted image. The amount and type of distortion will depend on the movement speed, type of the motion and depth of the scene. In general, the distortions can be arbitrary, but in practice, only several kinds of distortions appear such as skew, wobble, shrinking and extending.

The most common camera model used in computer vision methods is a perspective camera model which is a reasonable approximation for many consumer cameras and cell phones. Since those devices almost always contain image sensors with a rolling shutter, the perspective camera model will only be valid when the camera and the scene are not moving. It has been shown that RS image distortions can cause problems for standard computer vision methods such as Structure from Motion [47], visual SLAM [60] or multi-view dense stereo [94]. Therefore, having a special camera model for rolling shutter cameras is desirable.

The computing of an absolute camera pose is a key task in computer vision. Existing methods for Rolling Shutter absolute camera pose estimation have either special requirements on the type of data (e.g. video sequences [47], planar scenes [2]) or are computationally demanding [80]. A solution to the RS camera absolute pose that requires only a minimal number of points, works for still images, as well as video sequences, is desired for robust Augmented reality, Localization, and SfM applications. An efficient minimal point solution can be used in RANSAC to robustly verify the hypotheses and provide an estimate with the largest consensus. The efficiency is of the highest importance for real-time applications such as Augmented reality and Visual odometry.

Although state-of-the-art algorithms for rolling shutter absolute camera pose and bundle adjustment have shown promising results, to our best knowledge, no one has yet addressed the task of running a complete RS SfM pipeline on general unordered sets of images. This is an important topic since almost all images taken today, even the still ones, can be affected with rolling

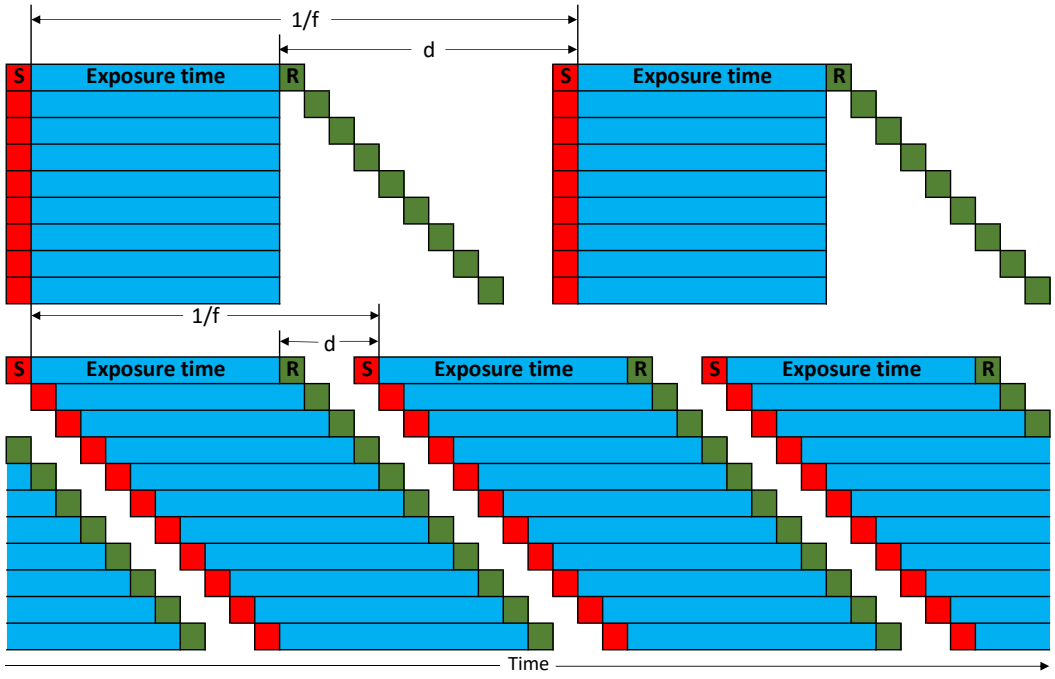


Figure 1.2: Diagram showing the differences in operation between global shutter (top) and rolling shutter (bottom) mechanisms. The time it takes from one frame to another is expressed as 1/f where f is the frame-rate. The inter-frame delay d is a time when the individual lines are not exposed. As you can see, the rolling shutter can achieve lower inter-frame delays and thus higher frame-rate. Conversely, with the same frame-rate, RS can have more prolonged exposure. Note that with RS some of the top lines of the next frame are exposed even when bottom lines of the previous frame are still exposed. With a large number of image lines and short exposures, it can happen, that the exposition of some lines of the next frame starts even before the exposition of the previous frame.

shutter distortion. Also, video sequences, where the rolling shutter is most apparent, are often not desirable to be processed frame by frame, because that is a massive computational load for longer sequences. The framerate available also could not be high enough for the interpolation used by [47]. Another issue is when combining data from different sources, where it is hard or impossible to enforce relationships between the camera poses and their motion. For these reasons, having SfM pipeline for rolling shutter images with no explicit constraints on the camera movement and temporal displacement is desirable.

Rolling shutter camera models describe the camera motion during image capture by a various number of additional parameters. This introduces extra dimensions of freedom to the model. For bundle adjustment, a key component of SfM, this can add new and undesired local minimum.

We observed in practice that the optimization tends to collapse into a degenerate solution which does not correspond to correct reconstruction in most of the cases (see Fig. 9.1). Although degenerate solutions have been studied for the case of perspective cameras, there has been no study for any of the rolling shutter camera models used today.

The availability of cheap and precise accelerometers and gyroscopes implies that almost every mobile phone is equipped with an inertial measurement unit (IMU). IMUs have also made their way into consumer cameras and allow for controlling and navigating robots as well as unmanned aerial vehicles (UAV). Due to this fact it is common to find both rolling shutter cameras and IMU's in the same device, making it attractive to utilize both together. In general, IMUs provide the "up vector", which is the orientation of the gravitational acceleration in the device frame, from which one can calculate the device rotation around two axes, e.g., pitch and roll. The accuracy of the orientation angular measurements is very high even for the low-cost IMUs. Using the IMU "up vector" information, we can eliminate some unknown parameters involved in the camera orientation estimation and thus make algorithms more efficient. Moreover, the IMU "up vector" information reduces the number of correspondences needed.

## 1.2 Two-camera calibration and synchronization

Many computer vision applications, e.g., human body modelling [102, 29], person tracking [31], pose estimation [35], robot navigation [20, 38], and 3D object scanning [98], benefit from using multiple-camera systems.

Calibrating multiple-camera systems can be decomposed into a task of calibrating couples of cameras and then chaining those calibrations into a single set of camera parameters for the entire system. Calibrating multiple pairs of cameras using a calibration pattern, such as the classical chessboard is very cumbersome and time-consuming due to the area that needs to be covered and the number of images that need to be taken. To alleviate this issue, many works focused on calibrating using a simpler target, e.g., a ball and modeling the spatiotemporal trajectories instead of the individual target poses.

In tightly-controlled laboratory setups, it is possible to have all cameras temporally synchronized. However, the applicability of multi-camera systems could be significantly enlarged when cameras might run without synchronization [45]. Synchronization is sometimes not possible, e.g. in the automotive industry, but even if it was possible, using asynchronous cameras may produce other benefits, e.g., reducing bandwidth requirements and improving the temporal resolution of event detection and motion recovery [30].

We propose an algorithm to robustly compute the geometry of two unsynchronized perspective cameras and simultaneously estimate the time shift between the image sequences taken by these two cameras. Obtaining the relative camera poses and time shifts we can successfully calibrate a unsynchronized multiple-camera system.

# 2 State of the Art

## 2.1 Rolling shutter

Several rolling shutter camera motion models were introduced in [81]. A projection function and optical flow equations have been formulated for fronto-parallel camera motion. The projection function was then used in bundle adjustment, comparing the results for BA with and without rolling shutter model. Significant improvement with the rolling shutter model is recorded on synthetic data, and a rule of thumb is derived for recommending when to use the rolling shutter model. Calibration procedure has been proposed to find out the rate of scan and the inter-frame delay using a flashing LED.

In [2] the authors took advantage of the rolling shutter effect, calculating the pose and motion of an object when the object shape is known, see figure 2.1. This is an analogous task to computing the absolute camera pose. They propose using a motion model based on Rodriguez formula for rotations and solve for the pose and motion parameters using non-linear optimization. For the initialization, they suggest an 8,5 point algorithm to compute the pose and motion parameters. The proposed algorithm requires a planar scene and the rotation motion is linearized using a first order Taylor expansion.

A polynomial solution to the RS absolute pose problem that is globally optimal was presented in [80]. It uses Gloptipoly [50] solver to find a solution from 7 or more points. Although it is possible to use as few as 7 points, usually more than 10 points are necessary to obtain satisfactory results. Authors show that the method provides better results than [2], but the runtime is in the order of seconds, which together with a larger number of points needed to obtain a good hypothesis make it impractical for typical applications such as RANSAC.

In [48], 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 VSLAM in [60] where the camera motion estimated from previous frames was used to compensate the RS distortion in the next frame prior to the optimization.

Authors of [95] present a minimal solution to the RS absolute pose for translational camera movement during capture. It is based on Gröbner basis automatic solver generator [65] and the authors propose a non-linear optimization on top of their solver to estimate also the camera rotation. The proposed solver does not take into account the camera rotation which is a dominant source of rolling shutter image distortions.

We presented the first minimal solution to the rolling shutter camera absolute pose problem in [5]. It uses the minimal number of six 2D to 3D point correspondences and the Gröbner

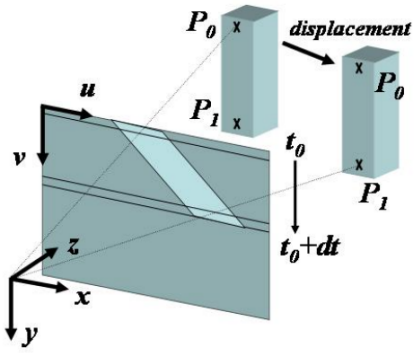


Figure 2.1: A moving object of known shape projects in an image with RS induced distortions(skew). Knowing the object shape allows us to calculate its pose and velocity from 2D↔3D correspondences. Image courtesy of [2]. This is an equivalent task to finding the absolute pose and velocity of the camera with respect to the object.

basis method to generate an efficient solver. The proposed R6P is based on the constant linear and angular velocity model as in [2, 80, 47] but it uses the first order approximation to both the camera orientation and angular velocity, and, therefore, it requires initialization of the camera orientation, e.g., from P3P [34]. Paper [5] 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.

In [7] we present a solver that eliminates the biggest drawback of [5], estimates the full camera orientation directly and therefore does not need the P3P initialization. Cayley parameterization is used to describe the camera orientation and the hidden variable resultant is used to simplify the equations for the solver generator [74]. The hidden variable resultant method is also used on the double linearized model equations in original R6P [5] and the more efficient solver is produced (300$\mu$s). The standalone R6P that uses only a single linearized model and does not need initialization of camera rotation is slower (1.4ms) but still usable in RANSAC and real-time applications.

We also present [8] much faster solution to [5] than the polynomial solvers, using iterative linear solvers, that make the equations completely linear by separating the unknowns and solving iteratively for their subgroups. This approach only requires solving of a small system of linear equations, that can be computed using fast QR decomposition. Equivalent results to [5] are achieved in as little as 20$\mu$s. This approach still requires a P3P initialization, since it uses the double-linearized model from [5].

In [66], a solution to the absolute pose problem using the "up vector" and two correspondences for calibrated cameras, or three correspondences for cameras with unknown focal length and radial distortion, has been presented. Minimal solutions to the calibrated relative pose prob-

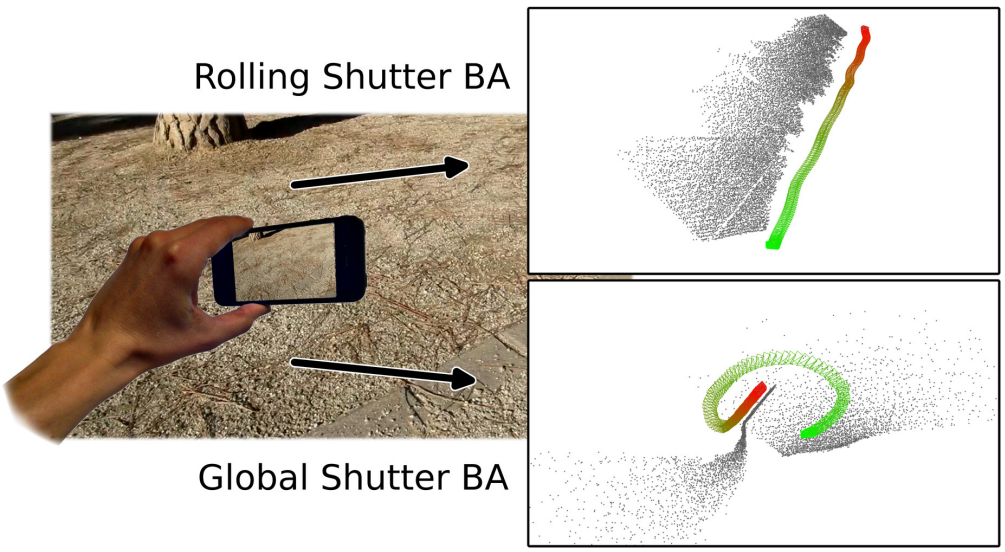


Figure 2.2: Structure from motion fails on rolling shutter images, e.g., videos from a hand-held smartphone. When a proper RS model is used, describing the camera motion during capture, SfM can recover the scene and camera geometry properly. Image courtesy of [47].

lem using three correspondences for two known orientation angles were proposed in [36, 59]. Relative pose for multi-camera systems with the aid of IMU has been presented in [76].

We utilized IMU measurements for RS absolute pose calculation in [6]. Using the "up vector" we were able to reduce the minimum number of required correspondences for RS absolute pose to five. The whole camera orientation can now be computed directly and no initialization is needed. The system of polynomial equations is easier to solve than in [5] and a solver running in $140\mu s$ is produced.

Relative pose problem for two rolling shutter cameras was investigated in detail in [27], pointing out that epipolar lines change to epipolar curves in the case of moving rolling shutter cameras. A generalized framework for describing both rolling shutter and push-broom camera geometry has been proposed. The authors derived essential matrices for either translational motion or both translational and rotational motion of the cameras. Both linear and non-linear algorithms to compute the relative pose were proposed, using a non-minimal number of correspondences in the linear case and the minimal number of points for the non-linear optimization.

## 2.2 Rolling shutter Structure from Motion

Structure from Motion (SfM) reconstructs the geometry of scenes from their images while simultaneously estimating camera poses and (some of) their internal parameters [44]. SfM has many practical applications in scene modelling, 3D mapping, and visual odometry [82, 103, 114].

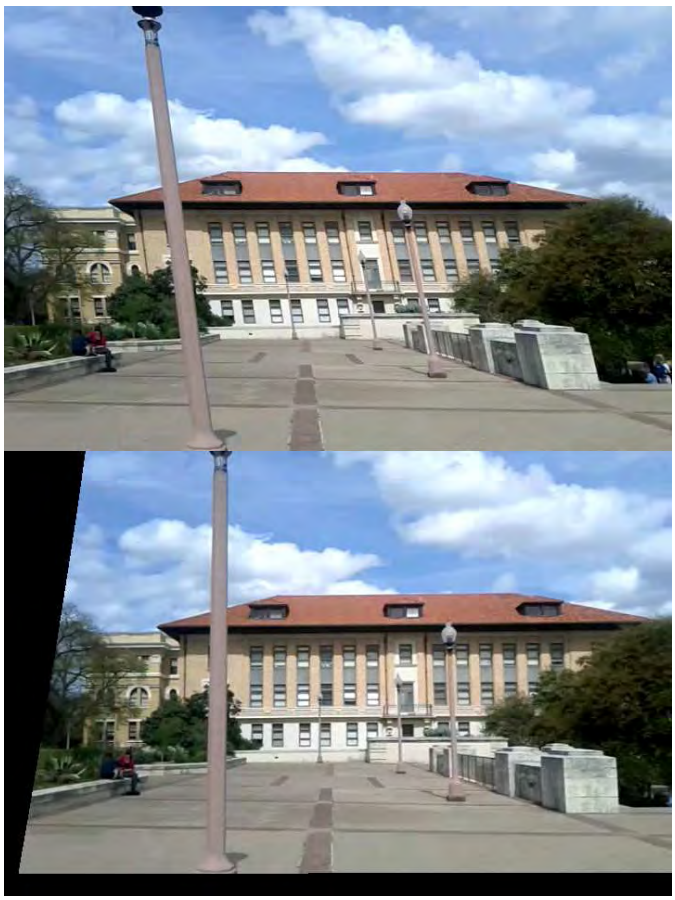

Figure 2.3: Rolling shutter distorted image (top) and rectified (bottom) using the probabilistic approach [56] that utilizes video sequences and IMU measurements to estimate the camera rotation. Image courtesy of [56].

Typical SfM considers perspective cameras, incrementally performs [103] and includes relative and absolute camera pose computation and bundle adjustment (BA) [107].

It has been observed [2, 48, 47] that image distortions caused by a moving RS camera can break 3D reconstruction and camera pose estimation down, see figure 2.2. To alleviate this problem, image rectification has been proposed to remove RS distortions [56, 92], see figure 2.3, and a structure from motion method for videos taken by rolling shutter cameras was developed [47]. It has been demonstrated [78] that using an RS model significantly improves the quality of mapping and tracking on mobile phones.

Authors of [48] addressed the problem of SfM from video sequences and presented specially adapted BA algorithm for rolling shutter videos [47]. In [60] an SfM pipeline for cell phone videos is presented using video sequences and fusion with inertial measurements. These works rely on the fact that video sequences contain images separated closely in time and space and therefore we can interpolate between camera poses. Authors of [3] presented a technique for

simultaneously estimating the shape and motion of an object with rolling shutter stereo pair and pointed out a degenerate case.

Previously mentioned work [81, 80, 5, 95] presents camera models that improve the precision of camera pose estimation under rolling shutter image distortions and which are viable candidates for BA optimization in SfM reconstruction.

In this thesis we address the problem of degenerate configurations in BA that uses these RS camera models, part of which was mentioned in [3] and which we more deeply investigated in [9]. We extended the analysis to a general case of two RS cameras and explained why BA with multiple non-constrained RS cameras sometimes produces flat scenes. A self-calibration approach to the critical configuration was presented in [54] where authors described the degeneracy in a specialized framework using affine cameras. They pointed out a solution which simply fixes some of the RS parameters in order to avoid the planar degeneracy.

## 2.3 Polynomial solvers

Camera geometry problems can be usually formulated in terms of multivariate polynomial equation systems. When camera geometry problems are solved as minimal problems, i.e. they are solved from a minimal number of input data and using all available constraints, the resulting systems of polynomial equations may be quite complicated. Minimal solvers are core parts of robust estimation schemes such as RANSAC [34] and therefore these solvers need to be fast.

The state-of-the-art approach for efficiently solving minimal problems is to create specific polynomial solvers [105, 63] for given problems that are based on Gröbner bases and action matrices [25, 63] or resultants [25, 67].

An approach to automatically generating solvers for systems of polynomial equations was presented in [65]. It is based on Gröbner bases theory [25] and has been since used to produce solvers for many important problems in computer vision e.g. [57, 16, 17, 68, 69, 71], robotics [70] or control theory [72]. Recently an improvement to the automatic generator [65] was presented in [74]. The proposed method exploits the inherent relations between the input polynomial equations and it results in more efficient solvers than [65]. The automatic method from [74] was later extended by a method for dealing with saturated ideals [75] and a method for detecting symmetries in polynomial systems [73].

Several approaches for optimizing Gröbner basis solvers with respect to stability and efficiency have been proposed recently. In [62, 65] and [83] the authors presented methods for optimizing the size of so-called elimination template matrices, i.e. matrices that are eliminated in Gröbner basis solver and are the core part of these solvers. Methods for improving numerical stability based on QR and SVD decomposition of template matrices were proposed in [19].

In [64] authors transformed elimination template matrices into a block diagonal form and in this way they sped up several minimal solvers.

A method for extracting univariate characteristic polynomial of the action matrix was proposed in [18]. The roots of the characteristic polynomial were found efficiently using Sturm-sequences [53], instead of computing the full eigendecomposition of the action matrix. This resulted in speedups of several important minimal problems.

Figure 2.4: A typical calibration of a stereo camera system's external and internal parameters. The procedure requires a lengthy process of acquiring images of a large planar pattern. The images need to be synchronized in time. Image courtesy of [13].

Gröbner basis method is more suitable for systems of polynomial equations with a small number of unknowns. A method for eliminating some unknowns from input equations and in this way simplifying resulting solvers was proposed in [71]. This method is based on elimination ideal theory [25].

## 2.4 Camera calibration and synchronization

Brown defined [14] a standard 8-parameter model for camera internal parameters that is frequently used today. Calibrating those parameters usually required images of objects with known 3D shape and a two-stage approach, where first the external parameters are estimated and then the internals [108]. A thorough investigation of the whole calibration process is given in [49], where authors propose a four-step approach.

A more convenient way that is widely used for calibrating cameras today is presented in [117]. The calibration requires only a known planar pattern, which is easily printed, and two or more images of that pattern from different viewpoints. The method computes the camera poses with respect to the planar pattern as well as the internal camera parameters, including radial distortion.

A method to calibrate camera internal parameters without any known pattern is called camera self-calibration. A classic work on this topic was done in [32], where authors present a method that calibrates the camera internals along with the external poses. It is basically a task similar to SfM. First, the epipolar geometry between adjacent views is estimated and then the Kruppa equations [44] are used to estimate the internal calibration.

To calibrate a multi-camera system using the planar pattern, one has to capture a significant amount of images of a pattern that is large enough to cover both fields of view simultaneously, see figure 2.4 and has to ensure that the images are taken at the exactly same time for all cameras [13]. Authors of the Kalibr calibration software [37] propose a unified framework for calibrating multiple sensors both spatially and temporally. Their formulation of the problem leads to a non-linear optimization problem, that jointly optimizes the calibration parameters and the temporal offset. However, an initial guess for the camera pose and time offset is needed.

Recently, authors of [111] presented a joint optimization of the camera intrinsics and extrinsics while performing a dynamic 3D reconstruction. The method takes multiple unsynchronized video streams and jointly optimizes the spatiotemporal scene and camera parameters using physics-based motion priors. While this work demonstrates the ability to synchronize and calibrate the cameras at the same time, it needs a proper initialization prior to the non-linear optimization. Finding a well-synchronized pair of images is still an issue.

Many video and/or image sequence synchronization methods are based on image content analysis [88, 21, 109, 26, 22, 30, 87, 90, 104], or on synchronizing video by audio tracks [101] and therefore their applicability is limited. Other approaches employed compressed video bitrate profiles [97] and still camera flashes [101].The methods differ in temporal transformation models. Often, time shift [88, 104, 109, 22], or time shift combined with variable frame rate [30, 87, 21], are used. The majority of previous work requires rigid sets of cameras. Notable examples of synchronization methods for independently moving cameras are [109, 22].

Many methods share a similar basis. A set of trajectories is detected in every video sequence using an interest point detector and an association rule or a 2D tracker. The trajectories are matched across sequences. A RANSAC based algorithm is often used to estimate jointly or iteratively the parameters of temporal and spatial transformations [30, 87, 21]. In [30], RANSAC is used to search for matching trajectory pairs in a filtered set of all combinations of trajectories in a sequence pair. The epipolar geometry has to be provided. The method [87] enables joint synchronization of $N$ sequences by fitting a single $N$-dimensional line called *timeline* in a RANSAC framework. The algorithm [21] estimates temporal and spatial transformation based on tentative trajectory matches.

Methods using exhaustive search to find the homography [104] and either fundamental matrix or homography [106] along with the time offset were presented. These are searching over the entire space of possible time shifts. Although in general applicable, they are computationally costly. The two most closely related works to ours are [86, 84] that jointly estimate two-view geometry together with time shift from approximated image point trajectories obtained from moving objects, see figure 2.5. In [86] estimated epipolar geometry or homography along with time shift using non-linear least squares, approximating the image trajectory by a straight line. The algorithm is initialized by the standard 7pt relative pose algorithm [44] and a zero time shift. Work [84] extended this approach by estimating the difference in frame rate and using splines instead of lines. Both the above works achieve good results only when given a good initialization, e.g., on sequences less than $0.5$ seconds time shift and with no gross matching errors because their least-square optimization approach is sensitive to outliers.

We investigated two-camera geometry for unsynchronized cameras in [4] and proposed solvers that compute either the fundamental matrix or homography along with an unknown time shift between sequences of images. Linear trajectory approximation was used as in [86], but we only use the minimal number of points and therefore are robust against outliers. An iterative algorithm was proposed that uses the new solvers to estimate large time shifts over tens of frames.

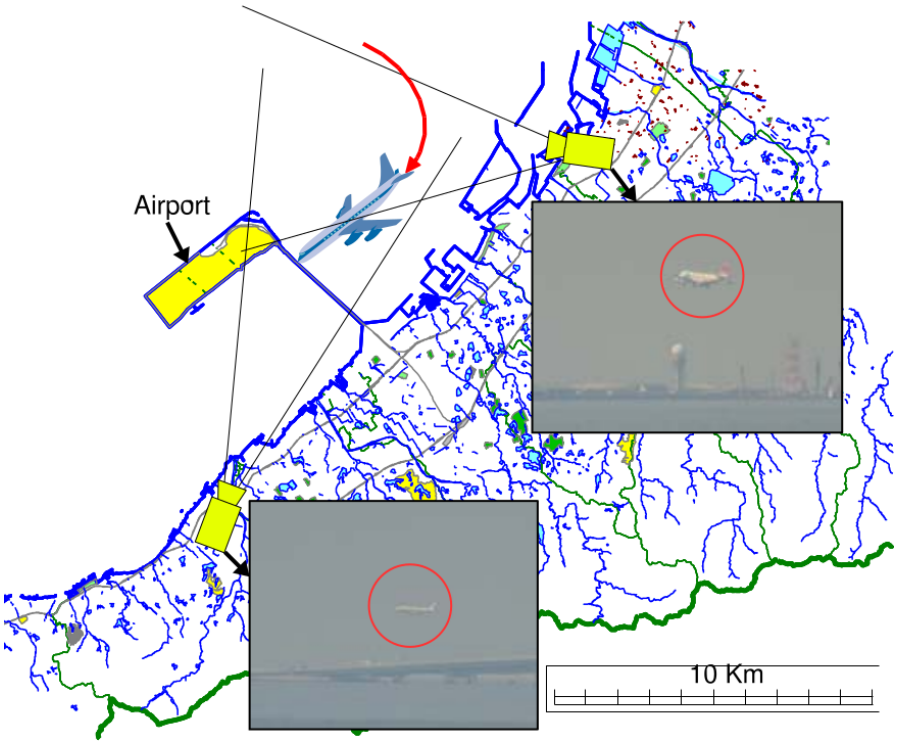


Figure 2.5: A two camera setup observing airplanes as they land and take off. Spatial and temporal calibration is needed to calculate the precise locations of the planes. Image courtesy of [86].

# 3 Contribution of the Thesis

The contribution of this thesis revolves around camera geometry that changes over time. Both rolling shutter and unsynchronized multi-camera systems present such cases when the cameras or objects in the scene move. A large portion of algorithms presented in this thesis are the minimal solutions to classical 3D vision problems adapted for situations when the camera or scene geometry varies over time. The remaining part deals mainly with problems arising from using different camera models than perspective and how to solve them. In particular, my contributions are the following:

1. **Rolling Shutter camera motion models**

   We investigate several motion models that can be applied in the case of moving rolling shutter cameras. Double-linearized model is proposed that leads to simpler polynomial equations and allows for the creation of efficient and stable minimal solvers for the camera absolute pose, presented in [5]. Another model, based on Cayley parameterization of rotation is presented that leads to slightly more complicated solvers that, however, don't require an initial camera orientation estimate as in the case of the double-linearized model. This solver will be presented in [7].

2. **Minimal non-iterative solutions to rolling Shutter camera absolute pose**

   We provide state-of-the-art solutions to the 6-point rolling shutter camera absolute pose problem (R6P), presented in [5]. A system of algebraic equations is formed, simplified and a solver is produced using the Gröbner basis method [65]. The solver based on double-linearized camera rotation model is fast (300$\mu$s) but requires an initial guess of the camera orientation which can be provided for example by P3P or obtained from IMU. The solver based on Cayley parameterization is a standalone solver that does not require any initialization, though it requires more time to produce the solution (1400$\mu$s). It is presented in [7].

3. **Minimal non-iterative solutions to rolling Shutter camera absolute pose with up-vector**

   A variant of the rolling shutter camera absolute pose problem which uses the information about up-vector (e.g. from the IMU) lets us produce a 5-point solution to the rolling shutter camera absolute pose (R5Pup), presented in [6]. The proposed solution is based on the linearized rolling shutter camera model used in [80], but requires only five 2D $\leftrightarrow$ 3D correspondences in contrast to seven in [80] and six in [5]. Using the vertical direction information we remove the requirement of prior initialization by P3P used in

R6P algorithm while providing even higher performance (140$\mu$s). We show that R5Pup solver works with data from IMU present in common smartphones and we present an RS aware Structure from Motion pipeline that uses R5Pup and handles imprecise up-vector measurements as well.

4. **Minimal linear iterative solutions to rolling shutter camera absolute pose**

   We present solutions which remove the slow performance drawback of R6P 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. Only a small system of linear equations is solved, which can be done using QR decomposition that is very efficient. In particular, one of the proposed solutions achieves comparable results to R6P in only 2 iterations, which takes around 20$\mu$s. The iterative linear solvers use the minimal number of points (six) as R6P, but we also propose a non-minimal 9-point solver that is non-iterative. This work will be presented in [8].

5. **Degeneracies in rolling shutter Structure from Motion and its solution**

   We show the degeneracies introduced by rolling shutter camera models and study them. The case of planar degeneracy which occurs most often in practice is explained and the reason why bundle adjustment always prefers this solution is given. We show that the presence of the degenerate solution is dependent on the relative alignment of the input images. Cases, where the scene can collapse into a plane for any number of cameras, are shown as well as situations where it is not possible. Our findings are backed by a number of both synthetic and real experiments that confirm the theory. We suggest a way to capture the images in practice such that the scene is reconstructed without any deformation. Again we verify the method on real data. This work is presented in [9].

6. **Two view geometry of unsynchronized cameras**

   We introduce practical solvers that simultaneously compute either a fundamental matrix or a homography and time shift between image sequences, and we propose a fast iterative algorithm that uses RANSAC [34] with our solvers in the inner loop to synchronize large time offsets. Our approach can accurately calibrate large time shifts, which was not possible before. Furthermore, our algorithms provide spatiotemporal calibration for two unsynchronized cameras capable of subframe synchronization precision. This contribution was presented in [4].

## 3.1 Thesis Outline

The rest of the thesis is organized as follows:

- **Chapter 4** introduces and briefly describes the key concepts that will be used and built upon further in this thesis.
- **Chapter 5** describes several different rolling shutter camera projection models that have been used in the literature or were proposed by us. A general formulation of the RS camera

projection is given. A discussion is provided considering the applicability of different RS camera motion models for BA and minimal solvers.

- **Chapter 6** formulates the problem of rolling shutter absolute pose and provides two minimal solutions. It describes how to form a system of equations such that efficient solvers can be produced using the Gröbner basis based automatic generator [65]. Both solvers are thoroughly tested on both synthetic and real data.

- **Chapter 7** extends the solvers presented in chapter 6 and formulates the problem of rolling shutter camera absolute pose with the use of up-vector. It describes how to incorporate the up-vector into the equations and produce much more efficient solver than R6P. We compare the performance against other viable solutions and provide a guide on how to implement the solver in incremental SfM.

- **Chapter 8** explores a different approach to the RnP problem and provides several iterative solvers that only require solutions to a small system of linear equations. An iterative solver is proposed that achieves an almost identical performance as the polynomial solvers but at a fraction of the time-cost. Also, a 9-point solver R9P is proposed that is non-iterative. A thorough investigation of the performance and comparison with R6P is provided.

- **Chapter 9** investigates the degeneracies that arise with the rolling shutter camera motion model. Mathematical analysis of the two-camera degeneracy is given and the conditions for three or more cameras degeneracy are empirically verified. Based on this analysis, we provide suggestion on how to capture images such that this degeneracy is avoided in general RS BA pipeline.

- **Chapter 10** deals with the problem of two view geometry for unsynchronized cameras. The problem is formulated using linear interpolation of the moving points trajectory. Two solutions for the fundamental matrix using either 8-points and Gröbner basis based solver or 9-points and generalized eigenvalues are presented. Furthermore, solutions for homography of unsynchronized cameras with time-shift are presented. An iterative algorithm built on top of the minimal solvers is proposed.

The individual chapters contain the work presented in our following publications.

| Chapter # | Publication |
|---|---|
| 6 | C. Albl, Z. Kukelova, T. Pajdla:<br>**R6P-Rolling shutter camera absolute pose**<br>CVPR 2015, Boston, Massachusetts, USA<br><br>C. Albl, Z. Kukelova, V. Larrson, T. Pajdla:<br>**Rolling Shutter Camera Absolute Pose**<br>*Transactions on Pattern Analysis and Machine Intelligence*, IEEE, submitted 2018 (1st revision - minor) |
| 7 | C. Albl, Z. Kukelova, and T. Pajdla:<br>**Rolling shutter absolute pose problem with known vertical direction**<br>CVPR 2016, Las Vegas, Nevada, USA |
| 8 | Z. Kukelova, C. Albl, A. Sugimoto and T. Pajdla:<br>**Linear solutions to rolling shutter absolute pose**<br>submitted to ACCV 2018 |
| 9 | C. Albl, A. Sugimoto, and T. Pajdla:<br>**Degeneracies in rolling shutter SfM**<br>ECCV 2016, Amsterdam, Netherlands |
| 10 | C. Albl, Z. Kukelova, A. Fitzgibbon, J. Heller, M. Smid and T. Pajdla:<br>**On the Two-View Geometry of Unsynchronized Cameras**<br>CVPR 2017, Honolulu, Hawaii, USA |

# 4 Key concepts

This chapter briefly describes the basic concepts and methods that are used and extended in the following chapters. We define the formulation of a camera, the tasks of camera absolute pose and relative pose estimation and the standard form of bundle adjustment that is used in computer vision problems.

## 4.1 Camera

The purpose of a camera is to capture the light reflected from objects and to form an image. We describe the process by mathematical models that suit the particular camera type. A most simple model is the pinhole camera model, which describes the original concept of *Camera Obscura*. The algorithms presented in this thesis will consider the perspective camera model which fits the majority of cameras without lens distortion or with the lens distortion incorporated in the camera model.

A most general form of camera projection function $\mathbf{f}$ takes the form of

$$\mathbf{u} = \mathbf{f}(\mathbf{a}, \mathbf{b}) \tag{4.1}$$

where $\mathbf{u}$ is the projected image point, $\mathbf{a}$ is the vector of parameters describing the camera and $\mathbf{b}$ is the vector containing the parameters of a 3D point. Suppose we use homogeneous coordinates and there is a 3D point with coordinates $\mathbf{X} = [X_1, X_2, X_3, 1]^\top$ that is projected into an image point $\mathbf{u} = [u_1, u_2, 1]$. The projection of this point can be written [44] as

$$\mathbf{u} = \boldsymbol{\pi}(\mathtt{P}\mathbf{X}), \tag{4.2}$$

where

$$\boldsymbol{\pi}\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}\right) = \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \\ 1 \end{bmatrix} \tag{4.3}$$

is the perspective projection in homogeneous coordinates. In following text we will sometimes also use a projection function of the form

$$\boldsymbol{\mu}\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}\right) = \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \end{bmatrix} \tag{4.4}$$

for situations where expressing $\mathbf{u}$ in non-homogeneous coordinates is more convenient. $\mathtt{P}$ is the $3 \times 4$ camera projection matrix [44] which can be decomposed as

$$\mathtt{P} = \mathtt{K} \left[ \mathtt{R} \;\middle|\; -\mathtt{R}\mathbf{C} \right], \tag{4.5}$$

where $\mathbf{C} \in \mathbb{R}^3$ is a camera projection center, $\mathtt{R} \in SO(3)$ is the camera orientation and $\mathtt{K}$ is the calibration matrix [44] containing the camera internal parameters. It is also possible to express $\mathtt{P}$ as

$$\mathtt{P} = \mathtt{K} \left[ \mathtt{R} \;\middle|\; \mathbf{T} \right], \tag{4.6}$$

where $\mathbf{T} = -\mathtt{R}\mathbf{C}$ is the camera projection center expressed in the camera coordinate system. Throughout the thesis, we will use whichever form of $\mathtt{P}$ is more suitable in a given situation. We call the camera calibrated when $\mathtt{K}$ is known and we can write

$$\mathtt{P} = \left[ \mathtt{R} \;\middle|\; -\mathtt{R}\mathbf{C} \right] = \mathtt{P} = \left[ \mathtt{R} | \mathbf{T} \right]. \tag{4.7}$$

This is a frequent case in computer vision since the camera internal parameters can be obtained by offline calibration and in many cases do not change over time.

## 4.2 Correspondences

As correspondences, we define the following two relations. An image correspondence, 2D↔2D, is the pair $\mathbf{u} \leftrightarrow \mathbf{u}'$ of projections of a single 3D point $\mathbf{X}$ into two images. A 3D↔2D correspondence, or scene to image correspondence, is the pair $\mathbf{X} \leftrightarrow \mathbf{u}$, where $\mathbf{u}$ is the projection of a 3D point $\mathbf{X}$ into the image.

## 4.3 Absolute camera pose

Computing absolute camera pose is a task of finding the camera parameters, i.e. $\mathbf{C}/\mathbf{T}$, $\mathtt{R}$ for the calibrated camera and also $\mathtt{K}$ for the non-calibrated camera from given 3D↔2D correspondences. Also called Perspective-n-Point (PnP), it is a fundamental problem in many computer vision tasks such as Structure from Motion, augmented reality, VSLAM, and visual localization.

The first solution to this problem was introduced by Grunert [41] and since then has been revisited many times [42, 10, 34]. Other work has focused on computing the absolute pose from over-determined systems defined by more than the minimally required number of correspondences [77, 89, 115, 91]. All of the previous work considers a perspective camera model , either calibrated or non-calibrated.

The rotation matrix $\mathtt{R}$ can be described by three parameters, and $\mathtt{K}$ contains five unknowns; therefore, the total number of parameters to estimate are 6 for the calibrated case and 11 for the non-calibrated one. Each scene to image correspondence gives us two equations and since equation (4.2) is homogeneous, three correspondences (P3P) are required to calculate the pose of a calibrated perspective camera and six (P6P) correspondences are required for a non-calibrated perspective camera as a minimum.

In this work, we present solutions to rolling shutter absolute pose that also solves for the motion parameters of the camera and requires the minimum of six scene to image correspondences (R6P).

## 4.4 Relative pose

Given two images, the relative pose describes the geometry of the camera pair that acquired them. A 3D point $\mathbf{X}$ projects to image point $\mathbf{u}_1$ in the first camera and $\mathbf{u}_2$ in the second camera. For each image correspondence $\mathbf{u}_1 \leftrightarrow \mathbf{u}_2$, there exists the epipolar constraint [44],

$$\mathbf{u}_2^\top \mathtt{F} \mathbf{u}_1 = 0. \tag{4.8}$$

The $3 \times 3$ matrix $\mathtt{F}$ is called the fundamental matrix and has rank two. It can be expressed in terms of the individual cameras parameters as

$$\mathtt{F} = \mathtt{K}_2^{-\top} \mathtt{R}_2 [\mathbf{C}_2 - \mathbf{C}_1]_\times \mathtt{R}_1^\top \mathtt{K}_1^{-1}. \tag{4.9}$$

Without loss of generality, since only the relative pose of the cameras with respect to another is important, we can assume that $\mathtt{R}_1 = \mathtt{I}$ and $\mathbf{C}_1 = [0, 0, 0]^\top$ and then

$$\mathtt{F} = \mathtt{K}_2^{-\top} \mathtt{R}_2 [\mathbf{C}_2]_\times \mathtt{K}_1^{-1}. \tag{4.10}$$

For calibrated cameras, the fundamental matrix $\mathtt{F}$ becomes the essential matrix $\mathtt{E}$ which in addition to being rank two has both eigenvalues identical. The essential matrix contains the relative orientation and translation between two cameras

$$\mathtt{E} = \mathtt{R}_2 [\mathbf{C}_2]_\times. \tag{4.11}$$

Since $\mathtt{E}$ is defined only up to scale, only the direction of the relative translation can be retrieved, not the magnitude.

In this work, we adjust the epipolar constraint equation (4.8) to approximate relationships between unsynchronized camera sequences.

## 4.5 Bundle adjustment

Bundle adjustment (BA) [107] is one of the key methods in 3D computer vision whose name comes from the ”bundles” of rays through which 3D objects are projected to the image. By ”adjusting” those rays we optimize the parameters of the entire scene, providing refined camera poses, internal parameters and 3D point locations (Figure 4.1). BA is widely used in SfM, visual SLAM, as a refinement step after estimating absolute pose or relative pose, and the triangulation of 3D points.

Since the projection function defining the rays is non-linear, the core of the BA method is a non-linear optimization. Algorithms for minimizing a non-linear function have been known since Newton and Gauss [85], more than three centuries ago. The method itself has been first

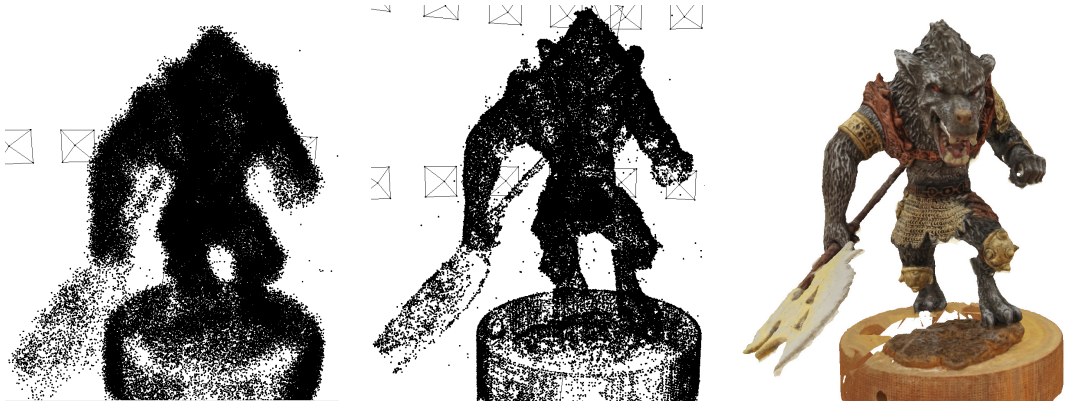

Figure 4.1: Bundle adjustment in action. After initial reconstruction (left), the refined parameters (middle) can be used to create a textured 3D model using dense reconstruction (right).

used in the field of Photogrammetry [15], and later on, received increasing popularity due to the availability of computational resources and effective algorithms. Despite its early appearance, BA still remains an interesting topic, which is apparent in the number of very recent publications regarding performance, scalability, robustness and other aspects of BA. Most of the state-of-the-art techniques that have been developed have been combined and implemented in the Google non-linear least squares solver Ceres [1].

Let us have $m$ cameras described by parameters $\mathbf{a} = (\mathbf{a}_1, \ldots, \mathbf{a}_m)$ and $n$ points described by parameters $\mathbf{b} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$. Also let $I$ be an index set, where $(i, j) \in I$ if point $i$ projects into camera $j$. The projection function is defined as:

$$\mathbf{u}_{ij} = f(\mathbf{a}_j, \mathbf{b}_i) \tag{4.12}$$

At last, we have a set of observed image feature coordinates $\hat{\mathbf{u}}_{ij}$ corresponding to each $(i, j) \in I$.

Starting with some estimate of the camera and 3D point parameters $\mathbf{p} = (\mathbf{a}, \mathbf{b})$, Bundle adjustment tries to find new parameters $\mathbf{p}^*$ such that

$$\mathbf{p}^* = \arg\min_{\mathbf{p}} c(\mathbf{p}) \tag{4.13}$$

where $c(\mathbf{p})$ is a cost function penalizing the errors between $\hat{\mathbf{u}}_{ij}$ and $\mathbf{u}_{ij}$, most commonly [15, 40, 24, 11, 113, 79, 55] in the form of

$$c(p) = \sum_{(i,j)\in I} ||\hat{\mathbf{u}}_{ij} - \mathbf{u}_{ij}||^2 \tag{4.14}$$

which is a non-linear least square minimization problem.

Detailed analysis of BA optimization methods, parameterizations, error modeling and constraints has been given in [107]. A standard algorithm of choice for solving equation (4.13) is

the Levenberg-Marquardt algorithm [44]. It is based on Gauss-Newton least square algorithm which iteratively seeks an update $\delta\mathbf{p}$ of $\mathbf{p}$ that reduces equation (4.14) by solving the so called normal equation

$$\mathtt{J}^\top \mathtt{J}\delta\mathbf{p} = \mathtt{J}^\top \mathbf{r} \tag{4.15}$$

where $\mathbf{J}$ is the jacobian of the projection function and $\mathbf{r}$ is a vector of residuals $\hat{\mathbf{u}}_{ij} - \mathbf{u}_{ij}$. Levenberg-Marquardt algorithm solves an augmented form of equation (4.15)

$$(\mathtt{J}^\top \mathtt{J} + \mu\mathtt{I})\delta\mathbf{p} = \mathtt{J}^\top \mathbf{r} \tag{4.16}$$

where the damping term $\mu$ allows for controlling the length and direction of the update step. The parameter vector $\mathbf{p}$ is updated only when a decrease in the cost function is achieved. If the cost did not decrease, $\mu$ is used to shorten the update step. The algorithm iterates until one of the stopping conditions are met.

In this thesis, we adapt BA by incorporating camera motion models into the projection function to accommodate for rolling shutter distortions. By introducing new parameters we increase the degrees of freedom which allows degenerate scene configurations to form. We analyze these degenerate configurations and show how they can be avoided.

## 4.6 Gröbner basis method

The minimal solvers presented throughout this thesis were created with the help of Gröbner basis automatic solver generators [65, 74]. Gröbner basis method is a very popular method for solving minimal problems in computer vision since these problems can often be described by systems of polynomial equations. Gröbner basis method is based on a polynomial ideal theory and it can provide fast and numerically stable solutions to such systems.

Let us have a system of $m$ polynomial equations

$$f_1(\mathbf{x}) = 0, \ldots, f_m(\mathbf{x}) = 0 \tag{4.17}$$

in $n$ unknowns $\mathbf{x} = (x_1, \ldots, x_n)$ which we want to solve and let this system have a finite number of solutions. The set of all polynomials that can be generated as polynomial combinations of the initial polynomials $f_1, ..., f_m$ defines an *ideal* $I = \{\Sigma_{i=1}^m f_i\, h_i \,|\, h_i \in \mathbb{C}\,[x_1, ..., x_n]\}$. In general, there are many different sets of generators that can generate an ideal and which all share the same set of solutions. However, there is a special set of generators called the reduced Gröbner basis w.r.t. the lexicographic ordering, which generates the ideal $I$ but is easy (often trivial) to solve [25] since this basis contains a univariate polynomial. Computing this lexicographic Gröbner basis and "reading off" the solutions from it is one of the standard methods for solving systems of polynomial equations [25].

Unfortunately, computing the Gröbner basis w.r.t. the lexicographic ordering for larger systems of polynomial equations, and therefore for most computer vision problems, may be computationally very expensive.

Fortunately, it is possible to construct a $G$ under a different ordering, e.g. the graded reverse lexicographic (grevlex) ordering, which is often easier to compute. This Gröbner basis $G$ is then

used to construct a special multiplication matrix $\mathtt{M}_p$ [25], also called the "action matrix". Let A be the quotient ring $A = \mathbb{C}[x_1, ..., x_n]/I$ [25] and $B = \left\{\mathbf{x}^\alpha | \overline{\mathbf{x}^\alpha}^G = \mathbf{x}^\alpha\right\}$ its monomial basis, where $\mathbf{x}^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} ... x_n^{\alpha_n}$ is a monomial and $\overline{\mathbf{x}^\alpha}^G$ is the remainder of $\mathbf{x}^\alpha$ after the division by a Gröbner basis $G$. Then the action matrix $\mathtt{M}_p$ is the matrix of a linear operator $T_p \colon A \to A$ performing multiplication by a suitably chosen polynomial $p$ in $A$ w.r.t. the basis $B$.

The action matrix $\mathtt{M}_p$ can be viewed as a generalization of the companion matrix used in solving one polynomial equation in one unknown [25], since the solutions to our system of polynomial equations (4.17) can be obtained from the eigenvalues and eigenvectors of this action matrix.

The output of automatic generators [65, 74] is a so-called elimination template that says which input polynomials should be multiplied with which monomials and eliminated to efficiently obtain such action matrix for the given problem. This means that the final Gröbner basis solvers perform only Gauss-Jordan elimination of the elimination template matrix and then they extract solutions to the input system from the eigenvalues and eigenvectors of the action matrix.

# 5 Rolling shutter camera models

## 5.1 Rolling shutter camera projection

Rolling shutter camera does not capture the entire frame at once, but line after line instead, usually from top to bottom. Since the theory is analogous regardless if we read out by row or column and whether we read out from top to bottom or vice versa, we will further focus only on the case where the image is being read out by rows from top to bottom. Consider that at time $t = 0$ an image row $r_0$ was captured. Then at time $t_c$ the index of a captured row $r$ can be calculated as

$$r = st_c - r_0 \tag{5.1}$$

where $s$ is the scan rate in lines per unit of time. The standard perspective camera projection of a 3D point with non-homogeneous coordinates $[X_1, X_2, X_3, 1]^\top = \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$ into an image point $\mathbf{u} = [c, r, 1]^\top$ is given by equation equation (4.2). In case of the rolling shutter camera which is moving during the capture the projection matrix is dependent on time and the projection becomes

$$\mathbf{u} = \boldsymbol{\pi}(\mathtt{P}(t_c) \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}), \tag{5.2}$$

where matrix $\mathtt{P}$ is a function of time

$$\mathtt{P}(t_c) = \mathtt{K} \begin{bmatrix} \mathtt{R}(t_c) & \mathbf{T}(t_c) \end{bmatrix} \tag{5.3}$$

Let us denote the time of capture $t_c$ and let the camera movement be described by translational velocity $\mathbf{t} = [t_x, t_y, t_z]^\top$ and an angular velocity $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]$.

We can write a linearized approximation to equation (5.3) for moving camera around a time $t = 0$

$$\mathtt{P}(t_c) = \mathtt{K} \begin{bmatrix} \left(\mathtt{I} + t_c \hat{\omega}^\top\right) \mathtt{R}(0) & \mathbf{T}(0) + t_c \mathbf{t} \end{bmatrix} \tag{5.4}$$

where

$$\hat{\omega} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} = \begin{bmatrix} \hat{\omega}_1^\top \\ \hat{\omega}_2^\top \\ \hat{\omega}_3^\top \end{bmatrix} \tag{5.5}$$

For further analysis we can assume, without loss of generality, that $\mathtt{K} = \mathtt{I}$, $\mathtt{R}(0) = \mathtt{I}$ and

$\mathbf{T}(0) = [0, 0, 0]^\top$ to make the equations easier. We can express $\mathtt{P}(t_c) \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$ as

$$\mathtt{P}(t_c) \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} = \begin{bmatrix} X_1 + t_c \hat{\omega}_1^\top \mathbf{X} + t_x t_c \\ X_2 + t_c \hat{\omega}_2^\top \mathbf{X} + t_y t_c \\ X_3 + t_c \hat{\omega}_3^\top \mathbf{X} + t_z t_c \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{5.6}$$

Next, we can notice that in order for a point to project into a specific line both equation (5.1) and equation (5.2) must be fulfilled for a specific line $r$ giving us

$$\frac{y}{z} = st_c - r_0 = r \tag{5.7}$$

which is a general rolling shutter camera constraint [81]. If we work in terms of the camera linear and angular velocities, we can derive a general projection model independent of $t_c$. In order to do that, we can substitute from equation (5.6) into equation (5.7) and solve for $t_c$.

$$\begin{array}{rcl} \frac{y}{z} & = & st_c - r_0 \\ \frac{X_2 + t_c \hat{\omega}_2^\top \mathbf{X} + t_y t_c}{X_3 + t_c \hat{\omega}_3^\top \mathbf{X} + t_z t_c} & = & st_c - r_0 \end{array} \tag{5.8}$$

and rewrite as

$$(s\hat{\omega}_3^\top \mathbf{X} + rt_z){t_c}^2 + (sX_3 - \hat{\omega}_2^\top \mathbf{X} - t_y - r_0 \hat{\omega}_3^\top \mathbf{X} - r_0 t_z)t_c - (X_2 + r_0 X_3) = 0 \tag{5.9}$$

which shows that in the most general form of motion, $t_c$ will be a solution of this quadratic equation. If we express $t_c$ and substitute it into equation (5.6), we can calculate the projections of a scene with a moving rolling shutter camera with known velocities.

From equation (5.9) we see, that not every type of movement will result in a quadratic equation for $t_c$, some special movements will result in a linear equation. To eliminate the second order term we need to get rid of $\hat{\omega}_3^\top = [-\omega_y, \omega_x, 0]$ and $t_z$, which means no rotation around $x$ and $y$ axes and no movement along the $z$ axis. Either of those moves causes the equation to be quadratic. We also see that under no movement, the equation reduces to $r = \frac{X_2}{X_3}$ which is the standard perspective projection.

### 5.1.1 Fronto-parallel motion

One special case of movement for which it is easy to show the projection function is the fronto-parallel motion for which

$$\mathbf{t} = \begin{bmatrix} t_x \\ t_y \\ 0 \end{bmatrix} \text{ and } \boldsymbol{\omega} = \begin{bmatrix} 0 & 0 & \omega_z \end{bmatrix} \tag{5.10}$$

Equation equation (5.9) reduces to

$$\begin{array}{rcl} (sX_3 - \omega_z X_1 - t_y)t_c & = & (X_2 + r_0 X_3) \\ t_c & = & \frac{X_2 + r_0 X_3}{sX_3 - \omega_z X_1 - t_y} \end{array} \tag{5.11}$$

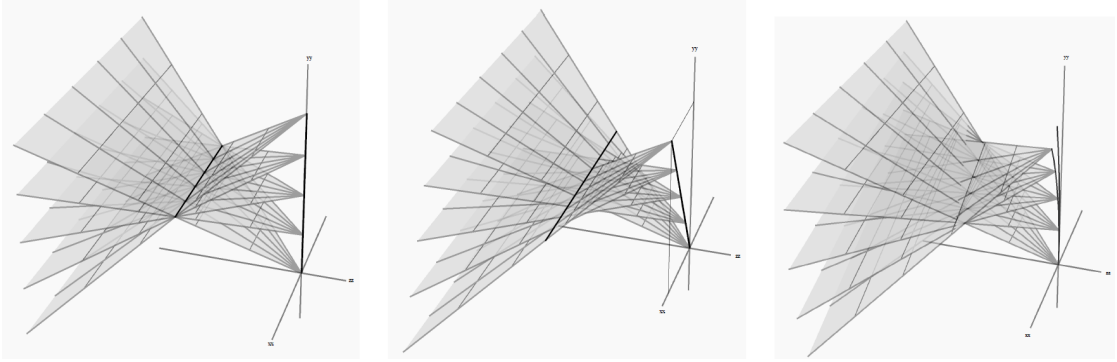

Figure 5.1: Left: when only $t_x \neq 0$ , middle when $t_x \neq 0$ and $t_y \neq 0$ and right when also $\omega_z \neq 0$ and the camera stops behaving like a cross-slit camera. Image courtesy of [81].

and using it in equation (5.6) and equation (5.2) we get

$$\begin{aligned}
\begin{bmatrix} c \\ r \\ 1 \end{bmatrix} &= \boldsymbol{\pi}\left(\mathtt{P}(t_c)\mathtt{X}\right) \\
&= \boldsymbol{\pi}\left(\begin{bmatrix} X_1 - t_c\omega_z X_2 + t_x t_c \\ X_2 + t_c\omega_z X_3 + t_y t_c \\ X_3 \end{bmatrix}\right) \\
&= \begin{bmatrix} \frac{X_1 - t_c\omega_z X_2 + t_x t_c}{X_3} \\ \frac{X_2 + t_c\omega_z X_3 + t_y t_c}{X_3} \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} \frac{X_1}{X_3} + \frac{t_c(t_x - \omega_z X_2)}{X_3} \\ \frac{X_2}{X_3} + \frac{t_c(t_y + \omega_z X_3)}{X_3} \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} \frac{X_1}{X_3} + \frac{X_2 + r_0 X_3}{sX_3 - \omega_z X_1 - t_y}\frac{t_x - \omega_z X_2}{X_3} \\ \frac{X_2}{X_3} + \frac{X_2 + r_0 X_3}{sX_3 - \omega_z X_1 - t_y}\frac{t_y + \omega_z X_3}{X_3} \\ 1 \end{bmatrix}
\end{aligned} \tag{5.12}$$

The projection is therefore a combination of perspective projection plus some additional term related to the motion. Notice that if the scan rate goes to infinity, the projection becomes perspective as for a global shutter camera.

We can invert projection equation (5.12) to obtain the rays which are projected to particular image points. For just a linear motion with $\omega_z = 0$ the camera will behave like a cross-slit camera, with the rays being incident with two lines in space, see figure 5.1.

## 5.2 Rolling shutter camera motion models

This section describes several camera motion models that can be used for solving the camera absolute pose and bundle adjustment with rolling shutter cameras. We discuss the feasibility of each model for both non-linear least square optimization used in BA and for creating fast minimal polynomial solvers using Gröbner bases based methods [25].

We are interested in describing the motion of the camera during image capture, which is usually a very short period of time, e.g. in orders of milliseconds or tens of milliseconds. Even though the camera motion can be arbitrary in general, due to the short time period reasonable motion approximations can be used.

For our purposes, without loss of generality, we will simplify the equations from previous section by setting the scan rate $s$ to 1 and the first scanned line $r_0$ to 0. From equation (5.1) we see that $t_c = r - r_0$ and we can parameterize our motion models in terms of the captured row. equation (5.3) now changes to

$$\mathtt{P}(r) = \mathtt{K}\begin{bmatrix}\mathtt{R}(r) & \mathbf{T}(r)\end{bmatrix}. \tag{5.13}$$

For the camera translation, a straight line constant velocity motion is generally used [94, 80, 81, 47, 2]. Good results achieved by previous works suggest that constant velocity is a sufficient approximation for the short time-span of a frame capture. We can write $\mathbf{T}(r)$ from equation equation (5.13) as

$$\mathbf{T}(r) = \mathbf{T} + r\mathbf{t} \tag{5.14}$$

where $\mathbf{T}$ is the camera center and $\mathbf{t}$ is the translational velocity.

Notice, that if $r = 0$ then $\mathbf{T}(r) = \mathbf{T}$ and the camera pose is equivalent to the perspective case. It is useful to add another constant parameter $r_p$ which will allow us to set for which row the RS camera model will reduce to the perspective one. The camera position is then

$$\mathbf{T}(r) = \mathbf{T} + (r - r_p)\mathbf{t} \tag{5.15}$$

For our purposes we choose $r_p$ to be the middle row of the image, which will be justified later.

## 5.3 SLERP model

To accommodate for camera rotation during frame capture we could interpolate between two orientations. A very popular method to interpolate rotations is SLERP [100], which was used in [47]. It works with quaternions and the formula to interpolate between two rotations represented by $\mathbf{q}_0$ and $\mathbf{q}_1$

$$SLERP(\mathbf{q}_0, \mathbf{q}_1, t) = \mathbf{q_0}\frac{\sin(\Omega - t\Omega)}{\sin\Omega} - \mathbf{q_1}\frac{\sin t\Omega}{\sin\Omega} \tag{5.16}$$

where $\Omega = \arccos\left(\mathbf{q_0}^\top \mathbf{q_1}\right)$ and $t \in (0, 1)$ is the interpolation parameter. 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, e.g. for the absolute pose. 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. This parameterization can be, however, utilized in optimization by bundle adjustment, such as in [47].

## 5.4 Euler vector model

Euler vector is co-directional with the rotation axis and the length of the vector is equal to the rotation angle. One can then write a rotation matrix associated with Euler vector $\mathbf{e}$ using the Rodriguez formula

$$\mathtt{R}(\mathbf{e}) = \mathtt{I} + \left[\frac{\mathbf{e}}{||\mathbf{e}||}\right]_\times \sin||\mathbf{e}|| + \left[\frac{\mathbf{e}}{||\mathbf{e}||}\right]_\times^2 (1 - \cos||\mathbf{e}||), \tag{5.17}$$

where $[\cdot]_\times$ is a skew symmetric matrix. We can treat $\mathbf{e}$ as a rotational velocity, which when multiplied by the row $r$ will give us a required camera orientation for each row. Further, to describe the camera rotation during image capture, we can separate the camera rotation matrix $\mathtt{R}(r)$ into two rotation matrices. $\mathtt{R}(\mathbf{v})$ describes the camera orientation whereas $\mathtt{R}(r\boldsymbol{\omega})$ is parameterized by the image row and describes the camera motion. The projection matrix equation (5.13) then becomes

$$\mathtt{P}(r) = \mathtt{K}\left[\mathtt{R}((r - r_p)\boldsymbol{\omega})\mathtt{R}(\mathbf{v}) \;\middle|\; \mathbf{T} + (r - r_p)\mathbf{t}\right] \tag{5.18}$$

where we can again by setting $r_p$ decide which row will have the pose equivalent to the perspective case.

Using Euler vector to describe camera rotation during capture yields the same advantages and drawbacks as using SLERP, i.e. multiplying the Euler vector by $r$ provides constant rotational velocity, but we have to deal with sine and cosine which makes it not practical for a minimal solver. It is, however, more practical than SLERP since we don't have to carry around two quaternions. The minimal solver we present in chapter 6 is based on a linearization of this model, which makes it easier to convert between those two and therefore easily initialize subsequent local optimization.

## 5.5 Cayley transform model

Another way to represent rotations is the Cayley transform [46]. For any vector $\mathbf{a} = [x, y, z]^\top \in \mathbb{R}^3$ there is a map

$$\mathtt{R}=\frac{1}{K}\begin{bmatrix} 1+x^2-y^2-z^2 & 2xy-2z & 2y+2xz \\ 2z+2xy & 1-x^2+y^2-z^2 & 2yz-2x \\ 2xz-2y & 2x+2yz & 1-x^2-y^2+z^2 \end{bmatrix} \tag{5.19}$$

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 $\tan\theta/2$, where $\theta$ is the rotation angle, thus 180 degree rotations are prohibited. We can again parameterize the camera orientation as in equation (5.18) by the image row $r$ to accommodate for camera rotation during capture, except that in this case we will not have a constant angular velocity.

Using this rotation model, we can prescribe the projection as a polynomial function. This allows for solving the absolute pose using Gröbner bases methods. However, the system of polynomial equations is too complicated to be solved efficiently, as will be discussed in the next chapter. This model is, as the previous ones, easy to incorporate into BA, where it can be used to refine the camera parameters. The disadvantage is the inability to describe 180 degrees and larger rotations, which can be slightly limiting in practical BA applications.

## 5.6 Linearized model

To simplify the problem we can linearize equation (5.18) $\mathtt{R}((r-r_p)\boldsymbol{\omega})$ around $r_p$ using the first order Taylor expansion and use Cayley parameterization for $\mathtt{R}(\mathbf{v})$ such that

$$\mathtt{R}(r\boldsymbol{\omega}) = \mathtt{I} + (r-r_p)[\boldsymbol{\omega}]_\times = \mathtt{I} + (r-r_p)\begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}, \tag{5.20}$$

which leads to

$$\mathtt{P}(r) = \mathtt{K}\left[(\mathtt{I} + (r-r_p)[\boldsymbol{\omega}]_\times)\mathtt{R}(\mathbf{v}) \;\middle|\; \mathbf{T} + (r-r_p)\mathbf{t}\right] \tag{5.21}$$

Such model, with the rolling shutter rotation linearized, was used in [80]. This function will deviate from a true rotation 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. It is practical to set $r_p$ as the middle row of the image, so that the error of the model is spread over the image symmetrically. In chapter 6 we will show how to produce a standalone solver for RS camera absolute pose using this model.

## 5.7 Double linearized model

Let us simplify the model even further by linearizing also the initial rotation matrix $\mathtt{R}_v$. We obtain

$$\mathtt{P}(r)=\mathtt{K}\left[(\mathtt{I}+(r-r_p)[\boldsymbol{\omega}]_\times)(\mathtt{I}+[\mathbf{v}]_\times)\middle|\mathbf{T}+(r-r_p)\mathbf{t}\right] \tag{5.22}$$

which leads to much simpler polynomial functions. 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 6.5 . In chapter 6 we will show how to create even faster solver for RS camera absolute pose than with equation (5.21), at the cost of having to obtain initial guess for the camera orientation.

# 6 Rolling shutter camera absolute pose

The computation of absolute camera pose using n 2D↔3D point correspondences under the rolling shutter effect (RnP) brings new challenges compared to PnP. For RS cameras, every image row will be captured at different time and hence at different positions when the camera is moving during the image capture. $\mathtt{R}$ and $\mathbf{T}$ from the projection equation equation (4.2) will therefore be functions of the image row $r$ being captured as described by equation (5.13) in the previous chapter.

We will further consider calibrated cameras, i.e. $\mathtt{K} = \mathtt{I}$. For i-th 2D↔3D correspondence we have

$$\lambda_i \mathbf{u}_i = \begin{bmatrix} c_i \\ r_i \\ 1 \end{bmatrix} = \mathtt{R}(r_i)\mathbf{X}_i + \mathbf{T}(r_i), \tag{6.1}$$

where $\lambda_i \in \mathbb{R}$ is an unknown scalar. In the previous chapter, we described several ways how to model $\mathtt{R}(r_i)$ and $\mathtt{T}(r_i)$. In the following sections we will describe the feasible models and how to use them to produce a fast, minimal solver for RnP.

We are interested in solvers from minimal number of correspondences because they provide an efficient way to estimate the model parameters via RANSAC [34] scheme. Since we have 12 unknown parameters $\mathbf{v}, \boldsymbol{\omega}, \mathbf{T}, \mathbf{t}$ and each correspondence gives two constraints, we need six points to solve the RS absolute pose and we will further call the solutions as R6P.

## 6.1 R6P formulation with Cayley transform model

We can prescribe equation (6.1) such that $\mathtt{R}(r_i)$ is a combination of two rotations written using Cayley transform as described in 5.5

$$\lambda_i \begin{bmatrix} c_i \\ r_i \\ 1 \end{bmatrix} = \mathtt{R}((r_i - r_p)\boldsymbol{\omega})\mathtt{R}(\mathbf{v})\mathbf{X}_i + \mathbf{T} + (r_i - r_p)\mathbf{t}. \tag{6.2}$$

to represent the camera initial orientation by $\mathbf{v}$ and the change of orientation during frame capture by $(r_i - r_p)\boldsymbol{\omega}$. This represents a rotation around the axis $\boldsymbol{\omega}$ which is close to uniform in angular velocity around $r_i = r_p$. Equation equation (6.2) is a rational polynomial and we must multiply it by $1 + x^2 + y^2 + z^2$ for both $\mathtt{R}(\mathbf{v})$ and $\mathtt{R}((r - r_p)\boldsymbol{\omega})$ 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 $\mathbf{T}$,$\mathbf{v}$,$\mathbf{t}$,$\boldsymbol{\omega}$ 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 very unstable. Due to these reasons we will not consider this model as feasible for our purposes.

## 6.2 R6P formulation with linearized model

Linearized RS motion model from section 5.6 gives us the following RS projection equation for each correspondence

$$\lambda_i \begin{bmatrix} c_i \\ r_i \\ 1 \end{bmatrix} = (\mathtt{I} + (r_i - r_p)[\boldsymbol{\omega}]_\times)\,\mathtt{R}(\mathbf{v})\mathbf{X}_i + \mathbf{T} + (r_i - r_p)\mathbf{t}. \tag{6.3}$$

That gives a system of degree three polynomials in 18 variables with 64 monomials. Simply inserting these equations into the automatic solver generator [65] yields a large, unstable solver. However, with some amount of manipulation, these equations can be transformed such that it is possible to find a reasonable Gröbner basis solver for this formulation. In section 6.4.3 we show how to simplify the equations and produce a viable polynomial solver.

## 6.3 R6P formulation with double linearized model

A further simplification of equation (6.3) is given in 5.7 and leads to following projection equation

$$\lambda_i \begin{bmatrix} c_i \\ r_i \\ 1 \end{bmatrix} = (\mathtt{I} + (r_i - r_p)[\boldsymbol{\omega}]_\times)(\mathtt{I} + [\mathbf{v}]_\times)\mathbf{X}_i + \mathbf{T} + (r_i - r_p)\mathbf{t} \tag{6.4}$$

which are simpler polynomial equations of degree two and 28 monomials. Still, clever equation rearrangement lets us produce a more efficient solver than simply inserting these equations in the automatic generator [65] directly. In section 6.4 we will show how to simplify equations equation (6.4) to make the computation more efficient and produce a fast polynomial solver.

## 6.4 R6P solvers

In previous chapter 5 we presented several rolling shutter camera models and in this section we show how to produce efficient solvers for the single and double linearized models suitable for RANSAC environment. To solve the polynomial equations of the two viable models (6.4,6.3) we use the Gröbner basis method [25]. 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 [25], see section 4.6.

### 6.4.1 Preparing the equations

We start with the first part that is similar for both solvers. The minimal number of 2D-to-3D point correspondences necessary to solve the absolute pose rolling shutter problem is six. For six point correspondence, both models (6.4,6.3) result in a quite complex system of $3 \times 6 = 18$ equations in 18 unknowns $(\lambda_1, \ldots, \lambda_6, \mathbf{v}, \boldsymbol{\omega}, \mathbf{T}, \mathbf{t})$. 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 (6.4,6.3) we first eliminate the scalar values $\lambda_i$ by multiplying all equations (6.4,6.3) from the left by the skew symmetric matrix

$$\begin{bmatrix} 0 & -1 & c_i \\ 1 & 0 & -r_i \\ -c_i & r_i & 0 \end{bmatrix}. \tag{6.5}$$

This leads to 18 equations, from which only 12 are linearly independent. For the double linearized model (6.4) they contain 22 monomials and 12 unknowns, i.e., the rotation parameters $\boldsymbol{\omega}, \mathbf{v}$ and the translation parameters $\mathbf{T}$ and $\mathbf{t}$.

For the single linearized model (6.3) we also need to multiply the equations by $(v_1^2 + v_2^2 + v_3^2 + 1)$ to get rid of the denominator coming from the cayley parameterization of $\mathtt{R}(\mathbf{v})$ equation (5.19). We need to keep in mind that $\mathbf{T}$ and $\mathbf{t}$ get multiplied as well and we obtain $\hat{\mathbf{T}} = \mathbf{T}(v_1^2 + v_2^2 + v_2^2 + 1)$ and $\hat{\mathbf{t}} = \mathbf{t}(v_1^2 + v_2^2 + v_2^2 + 1)$ and after solving for $\hat{\mathbf{T}}$ and $\hat{\mathbf{t}}$ we need to use $\mathbf{v}$ to obtain $\mathbf{T}$ and $\mathbf{t}$. The equations now contain 86 monomials in $\boldsymbol{\omega}, \mathbf{v}, \hat{\mathbf{T}}, \hat{\mathbf{t}}$.

The 12 linearly independent equations are linear in the unknown translation parameters $\mathbf{T}$ and $\mathbf{t}$ (or $\hat{\mathbf{T}}, \hat{\mathbf{t}}$ for the single linearized model). Therefore, they 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 $\boldsymbol{\omega}$ and $\mathbf{v}$ and substituting these expressions to the remaining six equations.

After the simplification we obtain a system of six equations in six unknowns and 16 monomials for the double linearized model and 80 monomials for the single linearized models. This system has 20 solutions for the double linearized model and 64 solutions for the single linearized model.

The system of six equations in six unknowns can be directly solved using the Gröbner basis method and the automatic generator of Gröbner basis solvers [65]. The Gröbner basis solver generated using the automatic generator [65] performs one G-J elimination of the 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 $\boldsymbol{\omega}$ and $\mathbf{v}$ are found from the eigenvectors of the multiplication matrix created from the rows of the eliminated template matrix.

Such a Gröbner basis solver for the R6P rolling shutter problem using the double linearized model was proposed in [5], it requires the G-J elimination of a $196 \times 216$ matrix and computing the eigenvectors of a a $20 \times 20$ matrix and it runs about 1.7ms. For the single linearized model the generated Gröbner basis solver was too large and unstable. In the next part we will describe how to simplify the problem further to produce better solvers, that are presented in [7].

### 6.4.2 R6P solver for the double linearized model

Here we present a much smaller solver to the R6P-2lin rolling shutter problem than the one proposed in [5]. First, note that six equations in six unknowns obtained from (6.4) by eliminating the scalar values $\lambda_i$ and the translation parameters $\mathbf{T}$ and $\mathbf{t}$ are bilinear in the rotation parameters, i.e. they are linear with respect to $\boldsymbol{\omega}$ and $\mathbf{v}$. Therefore, we can rewrite these six equations as

$$\mathtt{M}(\boldsymbol{\omega}) \begin{bmatrix} \mathbf{v} \\ 1 \end{bmatrix} = \mathbf{0} \tag{6.6}$$

where $\mathtt{M}(\boldsymbol{\omega})$ is a $6 \times 4$ matrix whose elements are linear polynomials in $\boldsymbol{\omega} = (w_1, w_2, w_3)$.

Since $\mathtt{M}(\boldsymbol{\omega})$ has a null vector, it must be rank deficient. Therefore, all the $4\times4$ sub-determinants of $\mathtt{M}(\boldsymbol{\omega})$ must equal zero. This results in $\binom{6}{4} = 15$ polynomial equations which only involve the rotation parameters $\boldsymbol{\omega}$. These fifteen polynomial equations can be written in a matrix form as

$$\mathtt{M}\mathbf{m} = \mathbf{0}, \tag{6.7}$$

where $\mathtt{M}$ is a $15 \times 35$ coefficient matrix and $\mathbf{m}$ is a $35 \times 1$ vector of monomials in three unknowns $w_1, w_2$ and $w_3$.

Note, that in this way we have eliminated additional three unknowns ($v_1, v_2$ and $v_3$) from our original system (6.4). Similar technique for eliminating unknowns from a bilinear system of polynomial equations was recently used in [110] to solve a minimal problem of estimating the motion of a multi-camera rig.

The system of fifteen polynomial equations in three unknowns equation (6.7) can be solved without the automatic generator of Gröbner basis solvers [65]. In this case, after performing G-J elimination of a coefficient matrix $\mathbf{M}$ we directly obtain a Gröbner basis for the ideal generated by the input fifteen polynomial equations. Therefore, to construct a $20\times20$ multiplication matrix it is sufficient to perform G-J elimination of a $15 \times 35$ coefficient matrix $\mathbf{M}$ equation (6.7). Then the solutions to the rotation parameters $\boldsymbol{\omega}$ are found from the eigenvectors of the multiplication matrix extracted from the eliminated $\mathtt{M}$. Solution to the remaining unknowns ($\mathbf{v}, \mathbf{T}, \mathbf{t}$) are found by back-substitution. The R6P-2lin solver constructed this way in C++ takes about 0.3 ms on a 2.5 GHz i7 CPU.

### 6.4.3 R6P solver for the single linearized model

For the single linearized model 6.3 we can use a similar approach as in the case of the double linearized model to obtain equations in only three unknowns. With the single linearized model the equations after the simplifications described in section 6.4.1 are linear with respect to $\boldsymbol{\omega}$. This time we can rewrite these six equations as

$$\mathtt{M}(\mathbf{v}) \begin{bmatrix} \boldsymbol{\omega} \\ 1 \end{bmatrix} = \mathbf{0} \tag{6.8}$$

where $\mathtt{M}(\mathbf{v})$ is again a $6 \times 4$ matrix which elements are now second degree polynomials in $\mathbf{v} = (v_1, v_2, v_3)$. Again we have $\binom{6}{4} = 15$ polynomial equations in the rotation parameters $\mathbf{v}$

coming from the $4 \times 4$ subdeterminants of $\mathtt{M}(\mathbf{v})$ that must be equal to zero. However, since the elements of $\mathtt{M}(\mathbf{v})$ are now quadratic in $\mathbf{v}$ this yields equations of degree 8.

These fifteen polynomial equations can again be written in a matrix form as

$$\mathtt{M}\mathbf{m} = \mathbf{0}, \tag{6.9}$$

where $\mathtt{M}$ is a $15 \times 165$ coefficient matrix and $\mathbf{m}$ is a $165 \times 1$ vector of monomials in three unknowns $v_1, v_2$ and $v_3$.

Unfortunately this time we introduced a two-dimensional family of false solutions when we eliminated $\boldsymbol{\omega}$. These solutions correspond to the first three columns of $\mathtt{M}(\mathbf{v})$ becoming linearly dependent. Then there will exist vectors in the nullspace of $\mathtt{M}(\mathbf{v})$ on the form,

$$\mathtt{M}(\mathbf{v}) \begin{bmatrix} \boldsymbol{\omega} \\ 0 \end{bmatrix} = \mathbf{0}, \tag{6.10}$$

which are not solutions to the original system. Studying these solutions revealed that they are all complex and satisfy

$$1 + v_1^2 + v_2^2 + v_3^2 = 0. \tag{6.11}$$

These do not correspond to valid scaled rotation matrices, e.g. the solution $\mathbf{v} = (i, 0, 0)$ corresponds to

$$\mathtt{R}(\mathbf{v}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & -2i \\ 0 & 2i & 2 \end{bmatrix}. \tag{6.12}$$

Fortunately the 15 polynomials were all divisible by equation (6.11). After dividing we have 15 equations of degree 6, which are only satisfied by the original 64 solutions.

Using the recent automatic generator technique from [74] we created a minimal solver for this system. The minimal solver uses an elimination template of size $99 \times 163$ to recover the $64 \times 64$ action matrix. After finding the solutions in $\mathbf{v}$ the remaining unknowns $(\boldsymbol{\omega}, \mathbf{T}, \mathbf{t})$ are found by back-substitution. The R6P-1lin solver constructed this way in C++ takes about 1.4 ms on a 2.5 GHz i7 CPU.

### 6.4.4 Pruning the solutions and improving performance

#### R6P-2lin

Usually only one of the 20 or 64 solutions of R6P-2lin is geometrically feasible, i.e., is real and of a reasonable values of parameters. Specifically, if we consider only reasonable values of the rolling shutter angular velocity $\boldsymbol{\omega}$ we can eliminate many solutions that are not feasible. Authors of [80] used the same linearization for $\boldsymbol{\omega}$ and showed that when $||\boldsymbol{\omega}|| > 0.05$ the model loses its accuracy. We decided to discard solutions with $||\boldsymbol{\omega}|| > 0.2$ which corresponds to angular velocity of approximately 11 degrees per frame. 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 sped up the process significantly by avoiding lots of work in model verification.

#### R6P-1lin

Unlike in the double linearized case, for the single linearized solver a significant portion of the computation time is spent computing the eigendecomposition of the 64x64 multiplication matrix $\mathtt{M}_m$. We can avoid the expensive eigendecomposition and use sturm sequences to compute the eigenvalues. The multiplication matrix is constructed in such way, that the eigenvalues are the solutions for the variable $v_3$. First we need to form the characteristic polynomial from the matrix $\mathtt{M}_m - \lambda\mathtt{I}$, e.g. by the Danilevsky method [28] and then find the eigenvalues using the Sturm sequences [53]. Moreover, using the Sturm sequence method we can search for solutions only in some feasible interval.

## 6.5 R6P-1lin - Getting close to the linearization point

As mentioned in section 5.2, the double linearized model will only be a good approximation when close to the linearization point. That is the case when $\mathtt{R}$ is close to $\mathtt{I}$. We can enforce this condition if we have an approximation $\mathtt{R}_a$ to $\mathtt{R}$. Then we can transform the 3D points as

$$\hat{\mathbf{X}}_i = \mathtt{R}_a\mathbf{X}_i \quad i = 1, \ldots, 6 \tag{6.13}$$

and replace $\mathbf{X}_i$ in equation (6.4) by $\hat{\mathbf{X}}_i$. Such solution $\mathtt{R}_l$ should then be close to $\mathtt{I}$ and we can obtain $\mathtt{R}$ as $\mathtt{R}_l\mathtt{R}_a$. 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 $\mathtt{R}_a$ using a standard P3P algorithm [42]. 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.

## 6.6 R6P-2lin - Staying in the domain

The domain of R6P-2lin is within the camera orientation angle $\alpha \in (-\pi; +\pi)$ and therefore there is a singular case of camera that we are not able to compute and that is a camera rotated by exactly 180 degrees. It is possible for most datasets to avoid this singular case, but even in the singular case there is a straightforward solution. For an arbitrary camera we can run R6P-2lin twice, once using $\mathbf{X}_i$ and once using $\hat{\mathbf{X}}_i = \mathtt{R}_a\mathbf{X}_i$ as in equation (6.13) where $\mathtt{R}_a$ is a rotation by 180 degrees around a random axis. This way we transform the space such that an arbitrary camera pose will fall into the domain of one of the two runs of R6P-1lin. In this case, we can reduce the performance increase for running R6P-1lin twice by realizing that we only need the results from the interval $\alpha \in [-\pi; +\pi]$ in each run.

## 6.7 Experiments

We conducted several experiments on synthetic as well as real datasets. The synthetic experiments were aimed at analyzing the properties of the double and single linearized rolling shutter camera models, which brings an interesting insight on how the solvers 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 solvers only to P3P solver [42], since to the best of our knowledge it is the only alternative with the same applicability. It is important to note that [47] can be used only on video sequences, [2] requires 9 co-planar points and [80] uses global optimization which is sensitive to mismatches and due to speed of Gloptipoly (in the orders of seconds) too slow for the RANSAC paradigm.

### 6.7.1 Synthetic data

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 the angle axis parameterization, because using the double linearized model or the single linearized model for generating data would allow R6P-2lin or R6P-1lin respectively to always find the exact solution. The image projections were obtained by solving equation (6.2) 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.

#### Handling the RS effect

The first experiment focused on varying the two rolling shutter parameters, i.e. the translational and angular velocity. The camera orientation is kept $\mathtt{R} = \mathtt{I}$ so we avoid the effect of the initial camera orientation linearization for R6P-2lin. We varied the angular velocity 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 Fig. 6.1. 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 28 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.

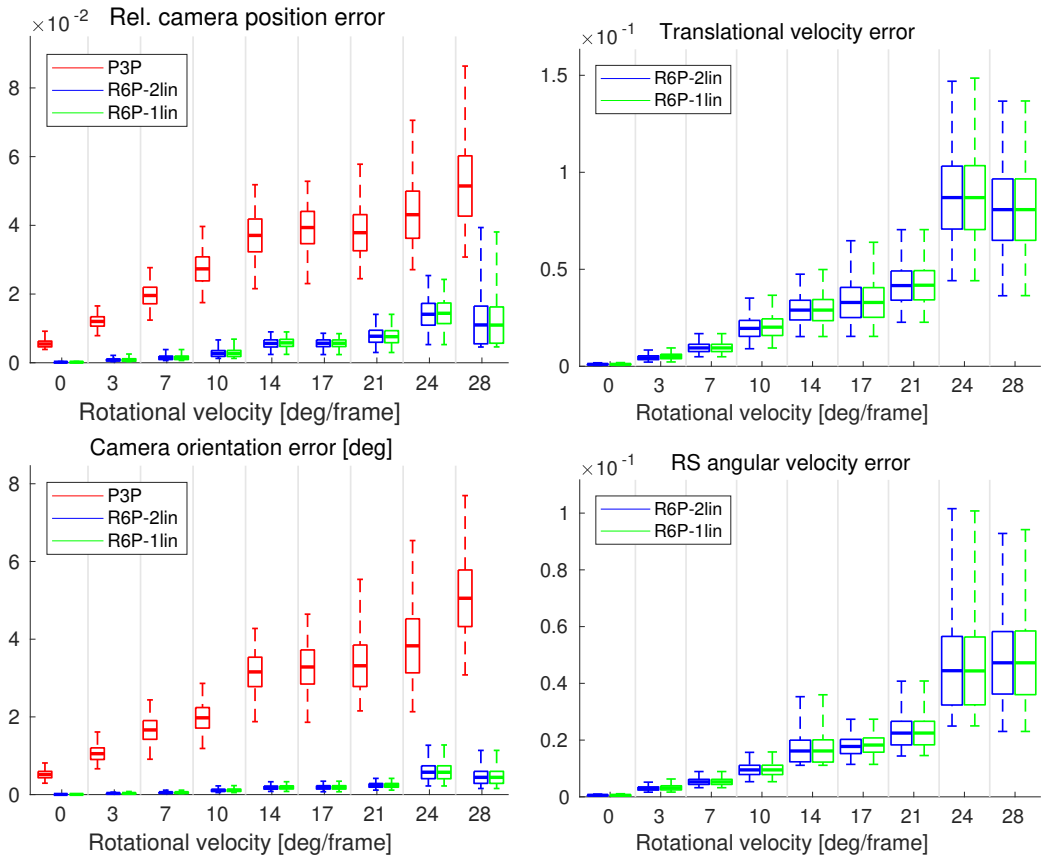


Figure 6.1: Experiment 1 - results of the estimated camera pose and velocity for varying RS motion, increasing the camera rotation as well as camera translation velocity. The camera orientation for R6P-2lin is kept at $\mathtt{R} = \mathtt{I}$ to avoid the effect of the linearized camera orientation.

### Effect of the double linearization in R6P-2lin

The second experiment was focused on finding out how well R6P-2lin behaves around its linearization point, i.e. when $\mathtt{R} \neq \mathtt{I}$. 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 $\langle 0; 15 \rangle$ degrees. Results are in Fig. 6.2 and they show that R6P-2lin 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 0.1 when the camera is rotated 15 degrees away from the linearization point. $\mathbf{T}$ and $\mathtt{R}$ are computed quite accurately, when $\mathtt{R}$ is within approximately 6 degrees from $\mathtt{I}$. It suggests that if we can use some standard non-RS method, such as P3P to find an initial $\mathtt{R}_0$ to align the

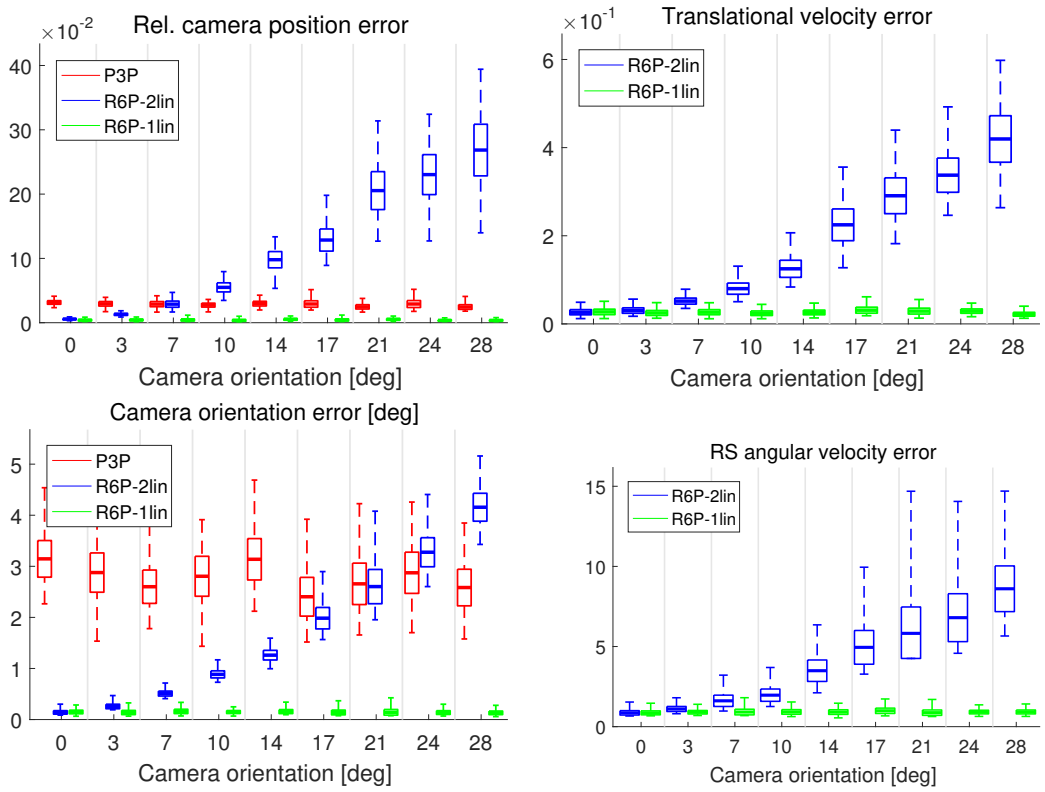


Figure 6.2: Experiment 2 - varying camera orientation, showing the effect of the double linearization of R6P-2lin. The camera angular and translational velocities are randomly chosen to not exceed 15 deg/frame and 0.15 respectively.

data, we can then apply our solver to get a more accurate camera pose. We tested this approach in experiment 3. R6P-1lin is not affected by the camera orientation as expected.

Results in Fig. 6.2 hint that the linearization is the key issue for R6P-2lin solver and that around 7 degrees of distance from $\mathtt{R} = \mathtt{I}$ 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 was capable of bringing the camera orientation within six degrees of the ground truth, even under large RS effect. At that point, if we apply R6P-2lin solver, the precision should improve significantly to values below 0.5 deg.

### Initialization of R6P-2lin by P3P

The purpose of experiment 3 is to verify that P3P can provide sufficient initialization for R6P-2lin. The camera orientation was chosen randomly and the RS parameters were increasing

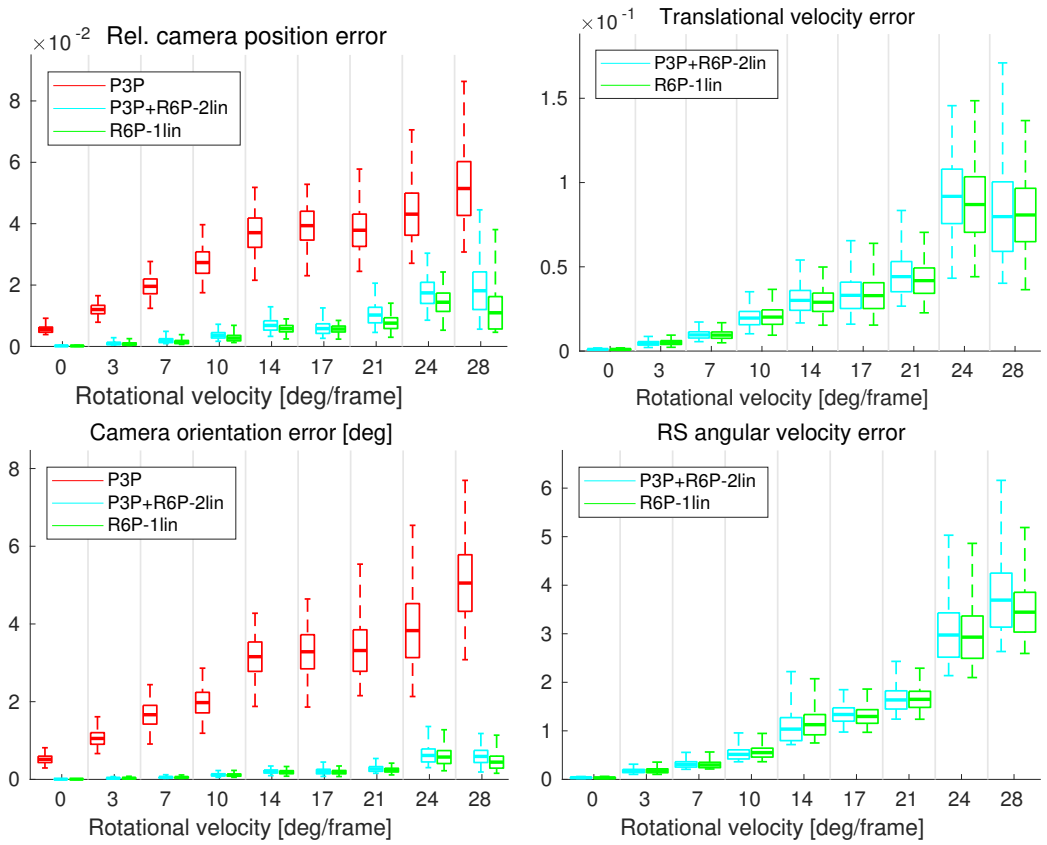


Figure 6.3: Experiment 3 - increasing camera motion and comparing the single linearized model (R6P-1lin) to the double linearized model (R6P-lin2) initialized by P3P. A significant improvement is made using R6P-2lin after being initialized with P3P. R6P-2lin initialized by P3P provides comparable performance to R6P-1lin but is outperformed by R6P-1lin on large RS effects because the initial orientation provided by P3P is not good enough.

as in experiment 1. We compared P3P, R6P-1lin and R6P-2lin initialized by P3P by using the orientation provided by P3P to rotate the scene as described in section 6.5. The results in Fig. 6.3 confirm our hypothesis. If the global P3P, or any other method, is able to compute the camera orientation within 6 degree error then R6P-2lin improves the solution to an average error below one degree. Interesting observation is that unlike in experiment 2 here the precision is significantly better for both camera center and orientation. It should also be noted, that R6P-1lin outperforms R6P-2lin as the RS effect increases, due to the deteriorating initialization provided by P3P.

### 6.7.2 Real data

We show two real world examples where our R6P solvers provide benefits. The first experiment is focused on estimating the camera absolute pose in the wild where the pose needs to be computed from image features possibly containing outliers. In a typical Structure from Motion application, the more 3D-2D matches are verified as inliers by the geometric camera model, the more 3D points will be reconstructed and the model better interconnected.

The second experiment shows an augmented reality scenario where known markers are placed in 3D space and they are detected automatically in the image. As camera absolute pose is computed using these markers, virtual objects can be placed into the scene. This represents a situation with a lot of camera movement and the need for a quick solver to run in real-time.

### 6.7.3 Structure from motion

In the first experiment we use datasets from [48] where an Iphone 4 camera was placed together with a global shutter Canon camera on a rig. Videos were taken when moving this rig by hand or walking around. We therefore have for each dataset two sets of images of the same scene. One set is with rolling shutter effect and one with global shutter.

#### 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 [103] 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. 2D correspondences being matched to wrong 3D points. 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%.

#### 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 first applied P3P to obtain $\mathtt{R}_a$ in equation equation (6.13), transformed the 3D points and then used our R6P-2lin solver as described in section 6.5. Since our data contained only few mismatches, 1000 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

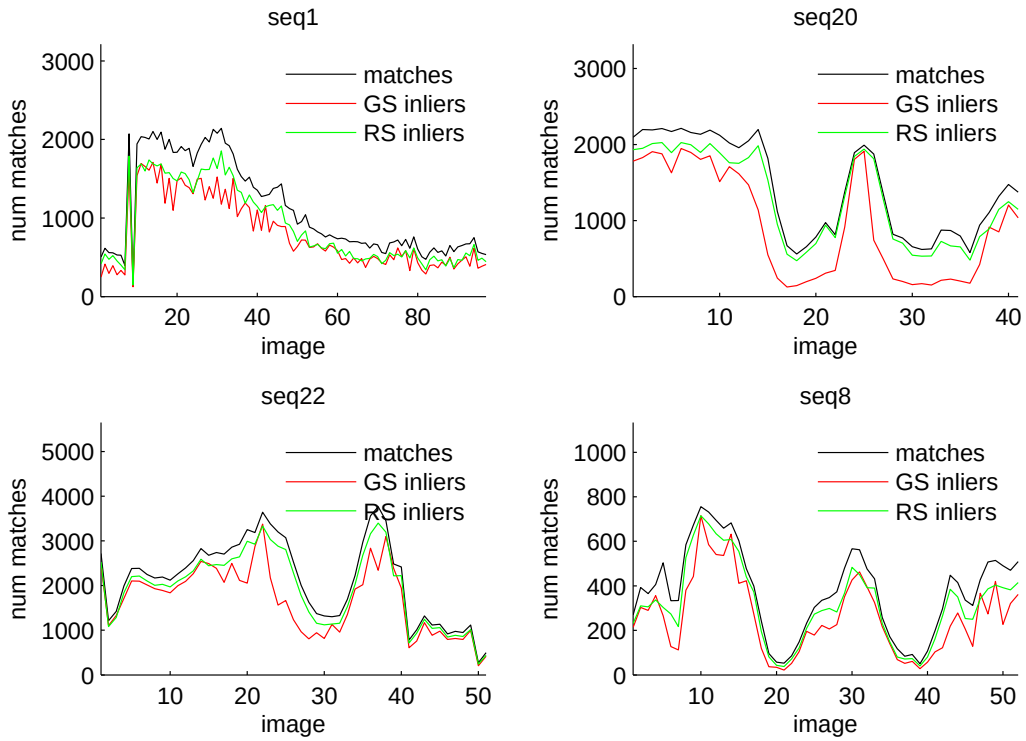


Figure 6.4: Examples of experiments on real data. Number of inliers after running 1000 rounds of RANSAC, averaged over 100 RANSAC runs. Number of 2D-3D matches from global shutter images to rolling shutter images are in black, number of inliers obtained by P3P are in red and number of inliers obtained by R6P-2lin are in green. The results are averaged over 100 runs to reduce randomness.

RANSAC runs. The inlier threshold in RANSAC was set to 0.002 of the image diagonal length which in this case was approximately 2 pixels. As it is seen in Fig. 6.4, R6P-2lin 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 and seq22 contained more camera motion and therefore show a larger gap between results of P3P and R6P-2lin.

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-2lin and P3P when the camera is still is comparable, although higher for R6P-2lin 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-2lin, in contrast,

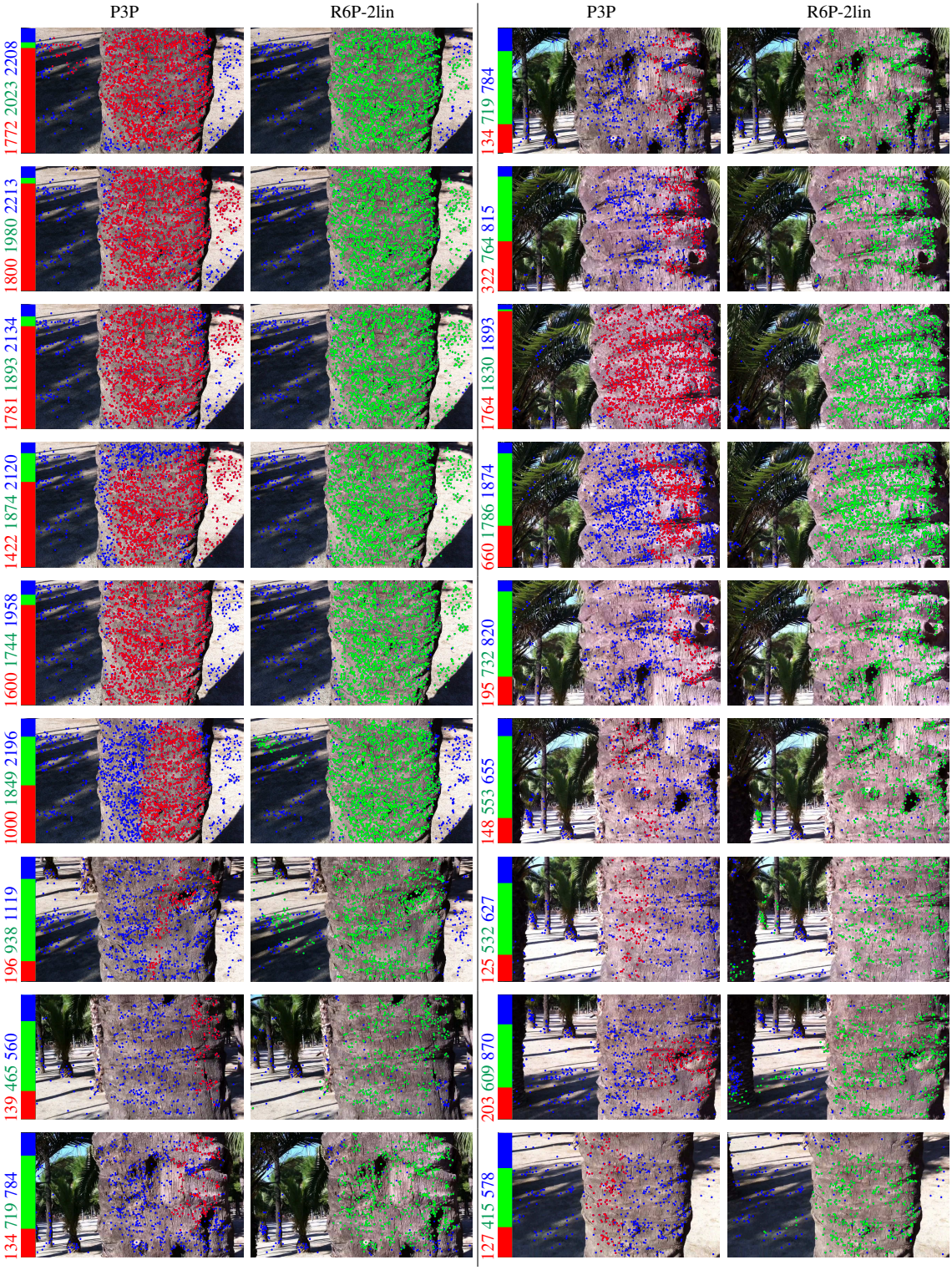


Figure 6.5: Results on dataset seq20, matched correspondences are in blue, inliers after RANSAC using P3P and R6P-2lin are in red and green respectively. The actual numbers of inliers are displayed on side of each image pair.

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-2lin to produce much better result. A detailed visualization of one of the results on seq20 is given in Fig. 6.5. We don't visualize the results of R6P-1lin because they were close to those of R6P-2lin and the detailed results can be found in table 6.1.

#### Very fast motion

We tested our solvers on heavily distorted RS images caused by fast camera motion that can arise in practice. A racing drone, carrying a GoPro camera and performing quick maneuvers is a good example of such data. We reconstructed the model of a building from approximately a hundred of still images from a classic digital camera combined with several images from the GoPro camera mounted on a drone which contained heavy RS distortion. We used a high quality state of the art SfM pipeline COLMAP [96] with the images being undistorted first.

The building was reconstructed well, but some of the RS images had clearly bad poses. In Figure 6.6 you can see that several RS images were reconstructed under ground. By using R6P-2lin we were able to recover better poses.

### 6.7.4 R6P and local optimization

An interesting question is whether one should use one of the two proposed R6P solvers or to locally optimize using one of the available Rolling-Shutter models starting from a P3P initialization. A popular approach called LO-RANSAC [23] uses the local optimization step after each accepted hypothesis in RANSAC. A camera model obtained by any method can be locally optimized using the points classified as inliers again using Bundle Adjustment (BA). We compared several meaningful approaches that represent possible practical use of the R6P algorithms and alternatives using local optimization. The key steps are following:

- P3P/R6P-2lin/R6P-1lin - RANSAC loop the corresponding solver
- LO-P - Local optimization inside RANSAC loop, followed by BA. Perspective model.
- LO-RS - Local optimization inside RANSAC loop, followed by BA. RS model.

We used LO-RANSAC and BA either with perspective or RS camera model with true rotation model from section 5.4. The local optimization step was implemented using Google Ceres [1].

We tested many possible practical combinations of algorithms and the results are shown in table 6.1 in terms of minimal and average number of inliers over the entire sequences. Methods in the first, fourth and fifth column represent the most straightforward use of the P3P and R6P solvers respectively with no local optimization. Method in the second column represents the best result that can be obtained using a perspective camera model, utilizing LO-P. Third column represent the case when we avoid using R6P solvers, but locally optimize using a RS model.

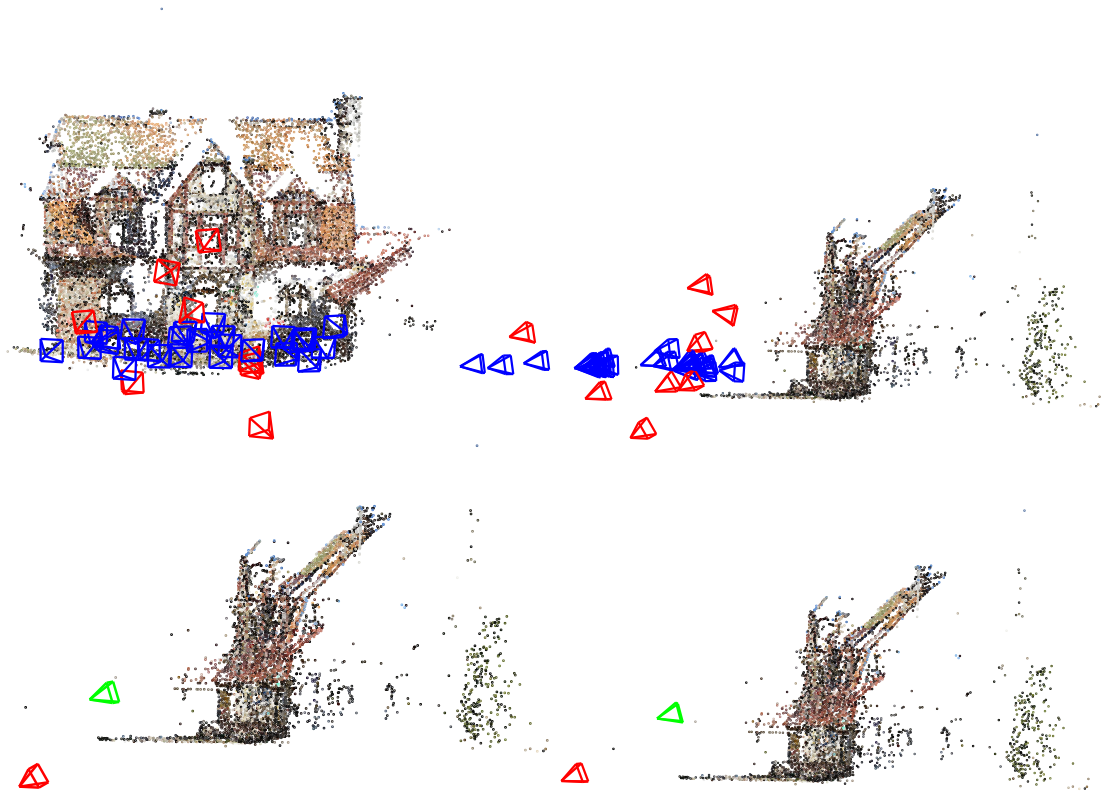

Figure 6.6: 3D reconstruction of a building from large number of GS images (blue camera poses) and several images containing high level of RS distortion (red camera poses). Several RS cameras were clearly reconstructed wrong, having the pose under ground. The poses estimated using R6P-2lin (green) are much more realistic.

Approaches in the sixth trough eighth column use R6P solvers with LO-RS, therefore utilizing everything available to achieve the best results.

We observe again that R6P greatly outperforms P3P in terms of inliers found. P3P with LO-RANSAC and BA using perspective camera model does not improve over P3P itself, signalizing that RS model is certainly needed for this type of data. A significant improvement is made when using P3P and optimizing with RS camera model. Still, it does not achieve the performance of R6P solvers without LO. On most of the datasets, R6P itself provides higher number of inliers as you can see in Fig. 6.7. It is mostly apparent in the minimum number of inliers, which indicates critical cases when the camera movement is large. This can be explained by the P3P not being able to identify a large enough set of inliers, that would provide a good set to optimize the RS model on.

There is a measurable difference between P3P + R6P-2lin and R6P-1lin in the favor of the latter, signalizing that P3P as an initialization of R6P-2lin can hinder the performance of the RS solver in situations where the RS effect is large and that R6P-1lin overcomes this problem.

Performance of R6P can be further improved by applying local optimization with RS model.

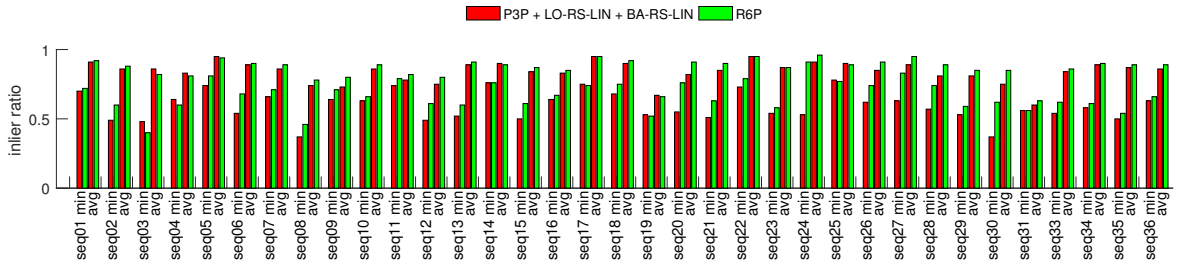


Figure 6.7: Comparison of the minimal and average ratio of inliers provided by R6P-2lin versus P3P with local optimization using RS model. Notice that R6P alone without local optimization is better on most datasets. In 28 datasets R6P-2lin provided higher minimal inlier ratio and in 25 dataset better average inilier ratio. In the remaining datasets, the R6P-2lin and locally optimized P3P provided almost identical results. The results correspond to columns 3 and 4 in table 6.1.

Using R6P solvers with subsequent local optimization with the true RS rotation model 5.4 provides the best performance across all datasets.

### 6.7.5 RANSAC threshold vs inliers

In this section we compare the number of inliers obtained by P3P and R6P-1lin with various RANSAC thresholds. As seen in the previous section on the real datasets P3P struggles to identify large portion of inliers on images with RS distortions compared to R6P-1lin with the same threshold. One could argue that increasing the threshold would allow P3P to capture more inliers, perhaps the same amount as in the case of R6P-1lin. To analyze this, we observed the number of inliers in each iteration of RANSAC under various thresholds. We chose dataset seq20 as a representative and average the results over 100 runs of RANSAC. The results in figure 6.8 show that in order to obtain the same percentage of inliers the threshold for P3P would have to be raised to 20 pixels, which is a significant increase compared to 2 pixels for R6P-1lin. Such large threshold could potentially lead to contamination by outliers.

### 6.7.6 Performance

To provide an idea about the performance of the proposed solvers in comparison to their alternatives we measured the time required by the sub-tasks in the RANSAC routine for each of the methods used in section 6.7.3. These sub-tasks include computing the camera pose, verifying the hypotheses and performing local optimization. Whereas computing the camera pose will always take approximately the same time, verification and local optimization will require different amount of time depending on the number of correspondences. The timings we show for verification and local optimization are for 1000 3D-2D correspondences which we chose as a reasonable representative case.

The timings in table 6.2 show that, while P3P is very fast compared to R6P, the verification is actually much more expensive than running P3P. The local optimimzation step is even more

| | P3P | | P3P LO-P | | P3P LO-RS | | P3P R6P-2lin | | R6P-1lin | | P3P R6P-2lin LO-RS | | P3P LO-RS R6P-2lin LO-RS | | R6P-1lin LO-RS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | min | avg | min | avg | min | avg | min | avg | min | avg | min | avg | min | avg | min | avg |
| seq01 | 0.53 | 0.77 | 0.48 | 0.76 | 0.70 | 0.91 | 0.72 | 0.92 | 0.72 | 0.91 | **0.77** | **0.94** | 0.76 | **0.94** | **0.77** | **0.94** |
| seq02 | 0.34 | 0.71 | 0.34 | 0.76 | 0.48 | 0.86 | 0.60 | 0.88 | 0.60 | 0.88 | **0.64** | **0.90** | **0.64** | **0.90** | **0.64** | **0.90** |
| seq03 | 0.39 | 0.76 | 0.40 | 0.77 | 0.60 | 0.89 | 0.72 | 0.89 | 0.73 | 0.89 | 0.75 | **0.92** | **0.76** | **0.92** | 0.75 | **0.92** |
| seq04 | 0.54 | 0.76 | 0.56 | 0.77 | 0.71 | 0.87 | 0.65 | 0.88 | 0.67 | 0.88 | **0.73** | **0.92** | **0.73** | **0.92** | **0.73** | **0.92** |
| seq05 | 0.37 | 0.82 | 0.38 | 0.82 | 0.67 | 0.94 | 0.83 | 0.94 | 0.83 | 0.95 | **0.86** | **0.97** | **0.86** | 0.96 | **0.86** | **0.97** |
| seq06 | 0.46 | 0.81 | 0.48 | 0.82 | 0.52 | 0.89 | 0.66 | 0.89 | 0.68 | 0.90 | 0.75 | 0.94 | **0.78** | **0.95** | 0.75 | 0.94 |
| seq07 | 0.44 | 0.69 | 0.44 | 0.70 | 0.65 | 0.85 | 0.72 | 0.89 | 0.70 | 0.89 | **0.76** | **0.92** | **0.76** | **0.92** | **0.76** | **0.92** |
| seq08 | 0.32 | 0.62 | 0.28 | 0.60 | 0.34 | 0.74 | 0.46 | 0.78 | 0.49 | 0.78 | **0.50** | 0.80 | **0.50** | **0.81** | **0.50** | 0.80 |
| seq09 | 0.28 | 0.42 | 0.29 | 0.43 | 0.58 | 0.72 | 0.70 | 0.80 | 0.71 | 0.80 | **0.78** | **0.84** | **0.78** | **0.84** | **0.78** | **0.84** |
| seq10 | 0.47 | 0.69 | 0.48 | 0.70 | 0.62 | 0.85 | 0.66 | 0.89 | 0.67 | 0.89 | **0.70** | **0.91** | **0.70** | **0.91** | **0.70** | **0.91** |
| seq11 | 0.57 | 0.64 | 0.58 | 0.65 | 0.72 | 0.76 | 0.78 | 0.81 | 0.81 | 0.82 | 0.81 | **0.85** | **0.83** | **0.85** | 0.81 | **0.85** |
| seq12 | 0.27 | 0.57 | 0.28 | 0.58 | 0.54 | 0.75 | 0.59 | 0.80 | 0.60 | 0.80 | **0.63** | **0.83** | 0.62 | **0.83** | **0.63** | **0.83** |
| seq13 | 0.41 | 0.74 | 0.42 | 0.74 | 0.53 | 0.89 | 0.60 | 0.91 | 0.63 | 0.92 | **0.65** | **0.93** | **0.65** | **0.93** | **0.65** | **0.93** |
| seq14 | 0.55 | 0.84 | 0.56 | 0.84 | 0.73 | 0.90 | 0.77 | 0.89 | 0.75 | 0.89 | **0.79** | **0.91** | 0.78 | **0.91** | **0.79** | **0.91** |
| seq15 | 0.46 | 0.68 | 0.46 | 0.68 | 0.51 | 0.84 | 0.64 | 0.87 | 0.62 | 0.87 | **0.65** | 0.89 | **0.65** | **0.90** | **0.65** | 0.89 |
| seq16 | 0.50 | 0.69 | 0.52 | 0.70 | 0.65 | 0.83 | 0.68 | 0.85 | 0.70 | 0.85 | 0.73 | **0.88** | **0.74** | **0.88** | 0.73 | **0.88** |
| seq17 | 0.65 | 0.78 | 0.66 | 0.80 | 0.75 | **0.96** | 0.74 | 0.95 | 0.76 | 0.95 | 0.78 | **0.96** | **0.79** | **0.96** | 0.78 | **0.96** |
| seq18 | 0.53 | 0.74 | 0.55 | 0.75 | 0.66 | 0.90 | 0.74 | 0.92 | 0.75 | 0.92 | **0.80** | **0.94** | 0.79 | **0.94** | **0.80** | **0.94** |
| seq19 | 0.48 | 0.63 | 0.49 | 0.64 | 0.53 | 0.68 | 0.51 | 0.66 | 0.53 | 0.67 | **0.57** | **0.71** | **0.57** | **0.71** | **0.57** | **0.71** |
| seq20 | 0.20 | 0.55 | 0.21 | 0.56 | 0.52 | 0.80 | 0.81 | 0.89 | 0.82 | 0.90 | 0.82 | **0.93** | **0.83** | **0.93** | 0.82 | **0.93** |
| seq21 | 0.31 | 0.59 | 0.32 | 0.60 | 0.51 | 0.84 | 0.64 | 0.90 | 0.63 | 0.90 | **0.67** | **0.92** | **0.67** | **0.92** | **0.67** | **0.92** |
| seq22 | 0.48 | 0.81 | 0.48 | 0.82 | 0.67 | 0.94 | 0.81 | 0.95 | 0.81 | 0.95 | **0.88** | **0.97** | **0.88** | **0.97** | **0.88** | **0.97** |
| seq23 | 0.37 | 0.73 | 0.38 | 0.74 | 0.52 | 0.87 | 0.57 | 0.87 | 0.59 | 0.88 | 0.62 | **0.90** | **0.63** | **0.90** | 0.62 | **0.90** |
| seq24 | 0.34 | 0.78 | 0.36 | 0.79 | 0.82 | 0.95 | 0.92 | 0.96 | 0.92 | 0.96 | **0.95** | **0.98** | **0.95** | **0.98** | **0.95** | **0.98** |
| seq25 | 0.47 | 0.74 | 0.48 | 0.75 | 0.79 | 0.90 | 0.76 | 0.89 | 0.77 | 0.90 | **0.82** | **0.92** | **0.82** | **0.92** | **0.82** | **0.92** |
| seq26 | 0.21 | 0.58 | 0.22 | 0.59 | 0.64 | 0.84 | 0.74 | 0.91 | 0.74 | 0.91 | 0.78 | **0.93** | **0.79** | **0.93** | 0.78 | **0.93** |
| seq27 | 0.26 | 0.61 | 0.26 | 0.61 | 0.63 | 0.87 | 0.83 | 0.95 | 0.83 | 0.95 | **0.85** | **0.96** | **0.85** | **0.96** | **0.85** | **0.96** |
| seq28 | 0.24 | 0.56 | 0.27 | 0.57 | 0.47 | 0.78 | 0.74 | 0.89 | 0.75 | 0.89 | **0.77** | **0.91** | **0.77** | **0.91** | **0.77** | **0.91** |
| seq29 | 0.37 | 0.67 | 0.38 | 0.67 | 0.50 | 0.81 | 0.60 | 0.85 | 0.59 | 0.85 | **0.64** | **0.88** | 0.62 | **0.88** | **0.64** | **0.88** |
| seq30 | 0.20 | 0.49 | 0.21 | 0.50 | 0.33 | 0.72 | 0.62 | 0.85 | 0.62 | 0.85 | **0.66** | **0.88** | **0.66** | **0.88** | **0.66** | **0.88** |
| seq31 | 0.41 | 0.50 | 0.42 | 0.51 | 0.53 | 0.59 | 0.55 | 0.63 | 0.56 | 0.63 | **0.60** | **0.67** | 0.58 | **0.67** | **0.60** | **0.67** |
| seq33 | 0.29 | 0.68 | 0.30 | 0.69 | 0.52 | 0.83 | 0.61 | 0.87 | 0.61 | 0.87 | **0.66** | **0.89** | **0.66** | **0.89** | **0.66** | **0.89** |
| seq34 | 0.32 | 0.79 | 0.33 | 0.80 | 0.73 | 0.94 | 0.87 | 0.96 | 0.87 | 0.96 | **0.89** | **0.97** | **0.89** | **0.97** | **0.89** | **0.97** |
| seq35 | 0.34 | 0.72 | 0.35 | 0.73 | 0.50 | 0.87 | 0.54 | 0.89 | 0.54 | 0.89 | **0.59** | **0.91** | 0.58 | **0.91** | **0.59** | **0.91** |
| seq36 | 0.40 | 0.75 | 0.40 | 0.76 | 0.51 | 0.88 | 0.56 | 0.89 | 0.56 | 0.89 | 0.58 | **0.91** | **0.60** | **0.91** | 0.58 | **0.91** |

Table 6.1: Comparison of different uses of P3P and R6P solvers. Table shows the minimum and average number of inliers found by the approaches described in section 6.7.4. R6P itself provides most of the time significantly better results than running P3P and LO-RANSAC and BA with RS model as visualized by the red and green colors in columns 3,4 and 5. The best results overall, marked by a bold font, are provided by R6P and subsequent local optimization with RS model.

expensive, an order of magnitude slower than running R6P-2lin. Depending on the application, these numbers will add to the total computation time which will depend on the algorithms used,

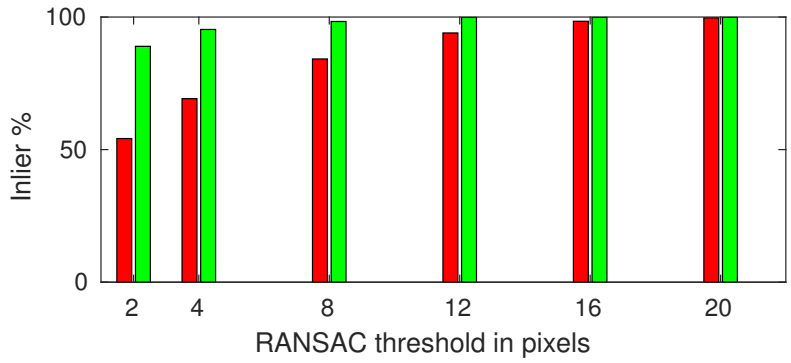


Figure 6.8: Average number of inliers found by P3P and R6P RANSAC using different thresholds.

| P3P | P-verification | R6P-2lin | R6P-1lin | RS-verification | LO-P | LO-RS |
|---|---|---|---|---|---|---|
| $0.5\mu s$ | $280\mu s$ | $300\mu s$ | $1400\mu s$ | $800\mu s$ | $9000\mu s$ | $17000\mu s$ |

Table 6.2: Average timings for different camera pose estimation tasks. 1000 correspondences assumed for the verification. The verification step also includes the average number of 2 hypotheses returned is for both P3P and R6P which makes a total amount of 2000 correspondences to be verified.

| **Method** | **Pose** | **Verif.** | **LO** | **Verif.** | **Total** |
|---|---|---|---|---|---|
| P3P | 0.5ms | 280ms | | | 280.5ms |
| P3P + LO-P | 0.5ms | 280ms | 90ms | 1.4ms | 373.3ms |
| P3P + LO-RS | 0.5ms | 280ms | 170ms | 4ms | 454.5ms |
| P3P + R6P-old | 300.5ms | 1080ms | | | 1380.5ms |
| P3P + R6P-old + LO-RS | 300.5ms | 1080ms | 170ms | 4ms | 1527.5ms |
| P3P + LO-RS + R6P-old + LO-RS | 300.5ms | 1080ms | 260ms | 5.4ms | 1645.9ms |
| R6P-new | 1400ms | 800ms | | | 2200ms |
| R6P-new + LO-RS | 1400 | 800ms | 170ms | 4ms | 2374ms |

Table 6.3: Average timings for different methods assuming 1000 rounds of RANSAC and 1000 correspondences per image. The amount of time spent by verification already includes the average number of hypotheses returned by the solvers.

number of RANSAC iterations, number of correspondences and number of local optimization steps. To give a better intuition about the total time complexity we provide a table 6.3 of runtimes of the different methods from section 6.7.3 based on the assumption that there are 1000 correspondences in the image, 1000 RANSAC steps and that local optimization is used 10 times during the LO-RANSAC procedure.

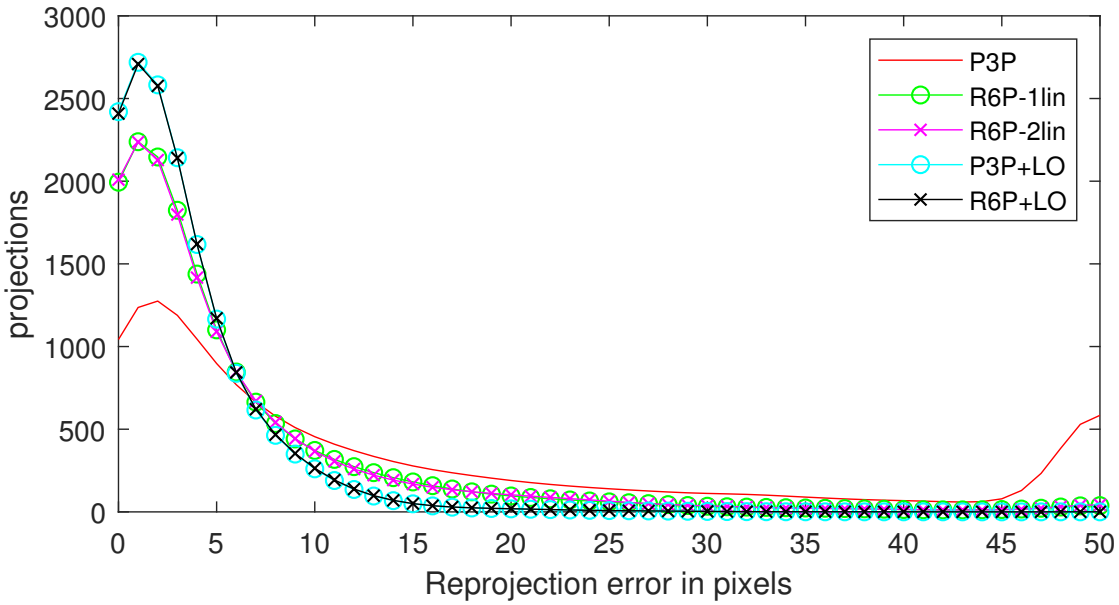


Figure 6.9: Reprojection errors on the detected aruco markers using the camera poses obtained by P3P, R6P-lin2, R6P-lin1 and P3P with subsequent local optimization (Bundle adjustment).

### 6.7.7 Augmented reality

We used the Aruco markers in a regular grid to provide an environment with known 3D-2D correspondences. A camera was used to take video of the scene, with random movement, simulating a person looking around. In total around 300 markers were present in the scene with approximately 120 markers detected in each image on average. On each frame we ran a 100 rounds of RANSAC to provide some robustness and we calculated the reprojection errors for the detected markers.

As soon as the camera started moving, the estimate provided by P3P started to be visually inaccurate. R6P provided a much more stable reprojection of the virtual objects. The farther the virtual object was from the detected markers from which the pose was computed, the more significant was the error of P3P. For an example of this effect, see figure 6.10.

To express this quantitatively. We calculated the reprojection error on the detected markers in each image. The results in figure 6.9 show that the pose computed by R6P provides overall much smaller reprojection errors. R6P provides almost as good performance as a P3P with subsequent local optimization.

## 6.8 Conclusion

In this chapter, we addressed the problem of absolute pose for cameras with rolling shutter. Two of the models presented in chapter 5 were found to be feasible to be used in an efficient polyno-

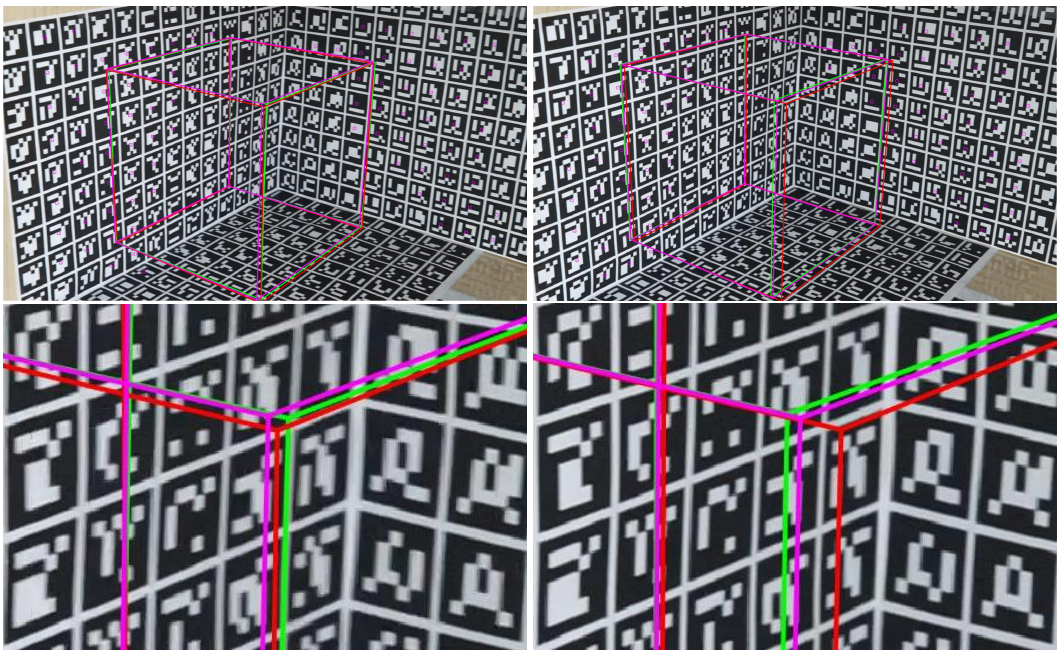

Figure 6.10: Placing virtual objects in the scene using absolute camera pose calculated from P3P (red), R6P (green) or P3P with subsequent local optimization (magenta). Two subsequent frames from a video sequence are dislplayed, showing the effect that RS camera motion has on the classic P3P algorithm. In the closeups you can see how the cube projected using P3P deviates (right) from its original pose (left) when the motion starts.

mial solver. Using these solvers, camera position, orientation, translational velocity and angular velocity can be computed using six 2D-to-3D correspondences. The R6P-1lin solver, based on Cayley parametereization of the camera orientation is a first self-sufficient minimal solution to the rolling shutter camera absolute pose problem. The R6P-2lin solver is faster, but uses a linear approximation to the camera orientation and therefore requires an initial guess of the camera orientation. We showed on synthetic as well as real datasets that standard P3P algorithm is able to provide this initialization. Further, having an initial guess on camera orientation, such as from an inertial measurement unit present in cellphones or UAV's, one could use the faster R6P-2lin solver directly. Both of the presented solvers improve on the precision of the camera absolute pose estimate when rolling shutter effect is present in the images, delivering average camera orientation error under half a degree (compared to six degrees for P3P) and relative camera position error under 2% (compared to 6% for P3P) even for large rolling shutter distortions in our synthetic experiments. The solvers were verified to work on real data, delivering increased number of inliers when using R6P over P3P in RANSAC. We evaluated the effects of non-linear refinement with both linearized and non-linearized rolling shutter rotation models and have shown that R6P provides higher number of inliers than P3P with subsequent non-linear refinement in most cases.

# 7 Rolling shutter absolute pose with known vertical direction

In this chapter, we present a solution to the rolling shutter absolute camera pose problem with known vertical direction (R5Pup). It is an extension of the 6-point solutions from previous chapters and it provides more practical absolute camera pose computation than R6P for modern cameras on mobile devices.

## 7.1 Problem Formulation

As with R6P int the previous chapter, we start with the RS camera motion model presented in chapter 5, which allows for a different camera pose for each captured image row. For each 3D↔2D correspondence i we can write

$$\lambda_i \mathbf{u}_i = \begin{bmatrix} c_i \\ r_i \\ 1 \end{bmatrix} = \mathtt{R}(r_i)\mathbf{X}_i + \mathbf{T}(r_i), \tag{7.1}$$

where $\lambda_i \in \mathbb{R}$ is an unknown scalar. For functions $\mathtt{R}(r_i)$, $\mathbf{T}(r_i)$ we use the model described in section 5.6, used also in [80], which assumes a linear approximation to the camera rotation during image capture. This model deviates from the true rotation with increasing rolling shutter effect. However, it has been observed [80, 5] that it is usually sufficient for the amount of rolling shutter rotation present in real situations. This gives the rolling shutter projection equation

$$\lambda_i \begin{bmatrix} c_i \\ r_i \\ 1 \end{bmatrix} = (\mathtt{I} + (r_i - r_p)[\boldsymbol{\omega}]_x)\,\mathtt{R}(\mathbf{v})\mathbf{X}_i + \mathbf{T} + (r_i - r_p)\mathbf{t}, \tag{7.2}$$

where $r_p$ is the image row where our model equals to a perspective camera.

In this chapter we assume that we are able to obtain the vertical direction of the camera, *i.e.* the coordinates of the world vector $[0, 1, 0]^\top$ in the camera coordinate system. This "up vector" can be obtained from vanishing points, e.g. [12], or from IMUs of mobile devices.

The "up vector" returned by the IMU gives us the rotation $\mathtt{R}_0$ of the camera around two axes, in this case the x-axis and the z-axis. Note, that IMU sometimes returns directly two angles $\psi_x$ and $\psi_z$ of the rotation

$$\mathtt{R}_0 = \begin{bmatrix} \cos(\psi_z) & -\sin(\psi_z) & 0 \\ \sin(\psi_z) & \cos(\psi_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi_x) & -\sin(\psi_x) \\ 0 & \sin(\psi_x) & \cos(\psi_x) \end{bmatrix} \tag{7.3}$$

of the camera around $x$ and $z$ axes.

With the known rotation matrix $\mathtt{R}_0$ around the $x$-axis and the $z$-axis, the only unknown parameter in the camera rotation matrix $\mathtt{R}(\mathbf{v})$ in (7.2) is the rotation angle $\psi_y$ around the vertical $y$-axis. Thus, we can write

$$\mathtt{R}(\mathbf{v}) = \mathtt{R}_0\mathtt{R}_y(\psi_y), \tag{7.4}$$

where $\mathtt{R}_0$ is the known rotation matrix (7.3) and

$$\mathtt{R}_y(\psi_y) = \begin{bmatrix} \cos(\psi_y) & 0 & -\sin(\psi_z) \\ 0 & 1 & 0 \\ \sin(\psi_y) & 0 & \cos(\psi_z) \end{bmatrix} \tag{7.5}$$

is the unknown rotation matrix around the $y$ axis.

This parametrization of the rotation matrix $\mathtt{R}_y$ contains trigonometric functions $\sin$ and $\cos$. To eliminate $\sin$ and $\cos$ and to obtain polynomial equations, we use the substitution $q = \tan(\frac{\psi_y}{2})$ for which there holds $\cos(\psi_y) = \frac{1-q^2}{1+q^2}$ and $\sin(\psi_y) = \frac{2q}{1+q^2}$. We can write

$$\mathtt{R}_y(\psi_y) = \frac{1}{1+q^2}\begin{bmatrix} 1-q^2 & 0 & -2q \\ 0 & 1+q^2 & 0 \\ 2q & 0 & 1-q^2 \end{bmatrix} = \frac{\hat{\mathtt{R}}_y(q)}{1+q^2}. \tag{7.6}$$

With this parameterization of the rotation, we can write the projection equation (7.2) as

$$\lambda_i \mathbf{u}_i = (\mathtt{I} + (r_i - r_p)[\boldsymbol{\omega}]_x)\,\frac{\mathtt{R}_0\hat{\mathtt{R}}y(q)}{1+q^2}\mathbf{X}_i + \mathbf{T} + (r_i - r_p)\mathbf{t}.$$

## 7.2 R5Pup solver

The R5Pup solver for absolute pose of a rolling shutter camera with known vertical direction from a minimal number of point correspondences starts with the projection equation (7.2) and the parametrization of the rotation (7.6). The scalar value $\lambda_i$ can be eliminated from equation (7.2) by multiplying it from the left by the skew symmetric matrix

$$\mathtt{S}_i = \begin{bmatrix} 0 & -1 & c_i \\ 1 & 0 & -r_i \\ -c_i & r_i & 0 \end{bmatrix}. \tag{7.7}$$

Moreover, to get rid of rational functions in the parametrization (7.6) we multiply projection equation (7.2) by the denominator $1+q^2$ to transform the equations into polynomials. To simplify the resulting system, we replace vector $(1+q^2)\mathbf{T}$ by vector $\hat{\mathbf{T}}$ of three new unknowns and the vector $(1+q^2)\mathbf{t}$ by vector $\hat{\mathbf{t}}$. This leads to the following matrix projection equation

$$\mathtt{S}_i\left((\mathtt{I} + (r_i - r_p)[\boldsymbol{\omega}]_\times)\,\mathtt{R}_0\hat{\mathtt{R}}_y\,(q)\,\mathbf{X}_i + \hat{\mathbf{T}} + (r_i - r_0)\hat{\mathbf{t}}\right) = \mathbf{0}. \tag{7.8}$$

This matrix equation results in three polynomial equations for each 2D $\leftrightarrow$ 3D point correspondence. However, since the skew symmetric matrix $\mathtt{S}_i$ has rank two, only two of these equations are linearly independent.

There are ten unknowns in equation (7.8), six unknown translation parameters $\mathbf{T}$ and $\mathbf{t}$, three unknown parameters in $\boldsymbol{\omega}$ and unknown rotation parameter $q$. Therefore, the minimal number of 2D $\leftrightarrow$ 3D point correspondences necessary to solve the absolute pose rolling shutter problem with known vertical direction is five.

For five point correspondences, the projection equation (7.8) results in 10 linearly independent equations in ten unknowns. These equations are linear in the unknown translation parameters $\hat{\mathbf{T}}$ and $\hat{\mathbf{t}}$. Therefore, these translation parameters can be easily eliminated from (7.8) by Gauss-Jordan (G-J) elimination of a matrix representing the input equations (7.8). Note that it is necessary to consider all 15 linearly dependent equations from (7.8) in this G-J elimination because different equations are linearly independent in different scene configurations.

Since six of the ten linearly independent equations of (7.8) are used for the elimination of $\hat{\mathbf{T}}$ and $\hat{\mathbf{t}}$, we are left with four equations in four unknowns $\boldsymbol{\omega}$ and $q$. Elements of the unknown vector $\boldsymbol{\omega}$ appear linearly in these four equations and thus the equations can be rewritten

$$\begin{bmatrix} p_{11}^{[2]}(q & p_{12}^{[2]}(q) & p_{13}^{[2]}(q) & p_{14}^{[2]}(q) \\ p_{21}^{[2]}(q) & p_{22}^{[2]}(q) & p_{23}^{[2]}(q) & p_{24}^{[2]}(q) \\ p_{31}^{[2]}(q) & p_{32}^{[2]}(q) & p_{33}^{[2]}(q) & p_{34}^{[2]}(q) \\ p_{41}^{[2]}(q) & p_{42}^{[2]}(q) & p_{43}^{[2]}(q) & p_{44}^{[2]}(q) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ 1 \end{bmatrix} = \mathtt{M}(q) \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ 1 \end{bmatrix} = \mathbf{0}, \tag{7.9}$$

where $p_{ij}(q)$ is a polynomial in $q$ and the upper index $[\cdot]$ denotes its degree. We know that the matrix equation equation (7.9) has a non-trivial solution if and only if the determinant of the $4\times4$ polynomial coefficient matrix $\mathtt{M}(q)$ is equal to zero. This determinant directly leads to a degree 8 polynomial equation in unknown rotation parameter $q$. Its solutions can be efficiently found using the Sturm sequences method [116]. After recovering up to eight real solutions for $q$, we can back-substitute them into equation (7.9) to recover $\boldsymbol{\omega}$ linearly. Finally, we back-substitute $q$ and $\boldsymbol{\omega}$ into (7.8) to linearly determine the translation vectors $\hat{\mathbf{T}} = (1+q^2)\mathbf{T}$ and $\hat{\mathbf{t}} = (1+q^2)\mathbf{t}$.

## 7.3 Experiments

In this section we analyze the performance of R5Pup. The properties of the solver behavior under different conditions were thoroughly evaluated on synthetic data. On the real data, R5Pup was compared against P3P and P5P algorithms which are the plausible alternatives used for perspective cameras.

We compared R5Pup to the following relevant algorithms for camera absolute pose estimation:

- **R6P** - a rolling shutter absolute pose from six points presented in [5],
- **P3P** - standard implementation based on [42],

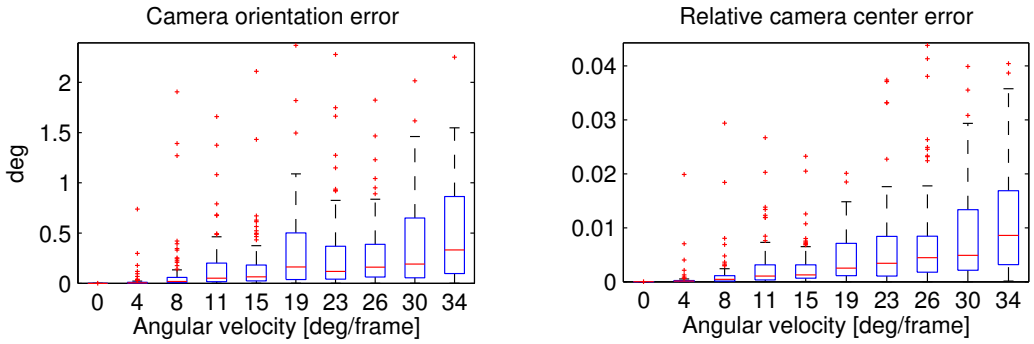


Figure 7.1: Results for varying camera angular velocity only.

- **P2Pup** - two-point perspective absolute pose solver using "up-vectop" presented in [66],
- **P5PLM** - PnP on five correspondences using iterative Levenberg-Marquardt optimization implemented in OpenCV,
- **EP5P** - PnP on five correspondences using the EPnP method of [77] implemented in OpenCV.
- **UPNP** - PnP on six correspondences using the UPnP method of [61].

### 7.3.1 Synthetic data

Experiments using synthetic data were aimed at showing R5Pup performance under different camera motions, presence of noise and erroneous estimates of the gravity vector. The data consisted of randomly placed points in a cube with side length 2 centered at the origin. Cameras were then placed randomly in the distance of $\langle 1; 3.3 \rangle$ from the origin. Cameras were calibrated with their field of view of 45 degrees. Since the solver returns up to 8 solutions we selected the one closest to the ground truth, since it would most likely be the one selected by RANSAC. Errors were measured in the camera orientation and position for all tested methods. For R5Pup we also evaluated the error in estimated angular velocity and translational velocity.

First, the algorithm was tested in the presence of camera motion and zero noise. Three cases were considered: (1) rotational movement only, (2) translational movement only and (3) both together. The camera motion was simulated using constant translational velocity and the Cayley parametrization model shown in [5]. For the case of rotational camera movement, the results in figure 7.1 show that the solver is able to deliver camera poses with relative position error under 1% and camera orientation error well under 0.5 degrees even for rapid camera rotation with more than 30 degrees per frame capture. The same results were observed for both camera rotational and translational movement, figure 7.2, showing that the camera translation movement does not have significant effect on the performance of R5Pup. The translation was varied up to 30% of the average distance of the camera from the 3D points. For pure translational movement in the absence of noise the solver produced exact results up to the machine precision which was

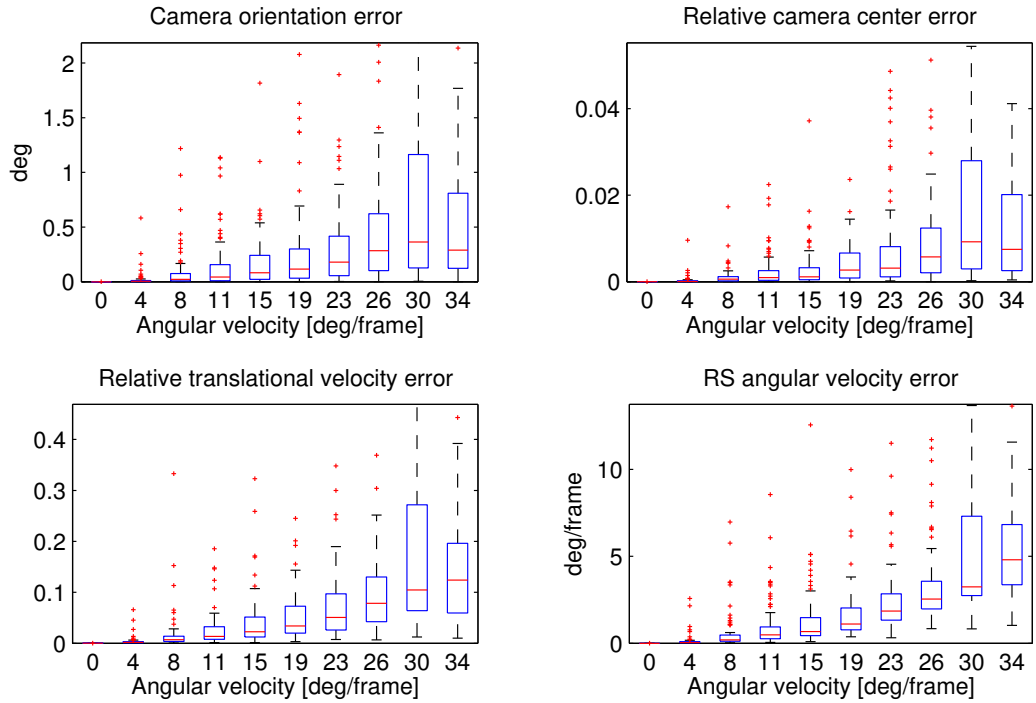


Figure 7.2: Results for varying both camera angular velocity and translational velocity.

expected since the model perfectly fits the data. This holds also for the case of zero camera rotation velocity in both figures 7.1 and 7.2.

Next, the susceptibility of R5Pup to noise was analyzed. The results in figure 7.4 show that noisy measurements in the presence of rolling shutter distortion do not significantly affect the performance of perspective camera methods and only slightly influence the result of R5Pup and R6P. We account this to the fact that the distortion caused by rolling shutter acts itself as noise of high magnitude for the perspective camera absolute pose algorithms and the noise added by imprecise feature detection or camera quantization is negligible compared to the RS effect.

The important question is, how does the error in the vertical direction estimation influence the results. Today even low cost IMU's can provide the gravity direction with accuracy under 0.5 degrees. However, during larger camera movements which cause significant RS image distortions we expect the gravity direction error to be higher. Therefore, we tested errors up to two degrees. The rotational velocity was set to 20 degrees per frame and relative translational velocity as 10%. It is clear that the precision of the IMU is critical for R5P. Results in figure 7.5 show that R5Pup outperforms other methods in the camera orientation estimation up to two degrees of angular gravity direction error and in the camera center estimation up to one degree angular gravity direction error.

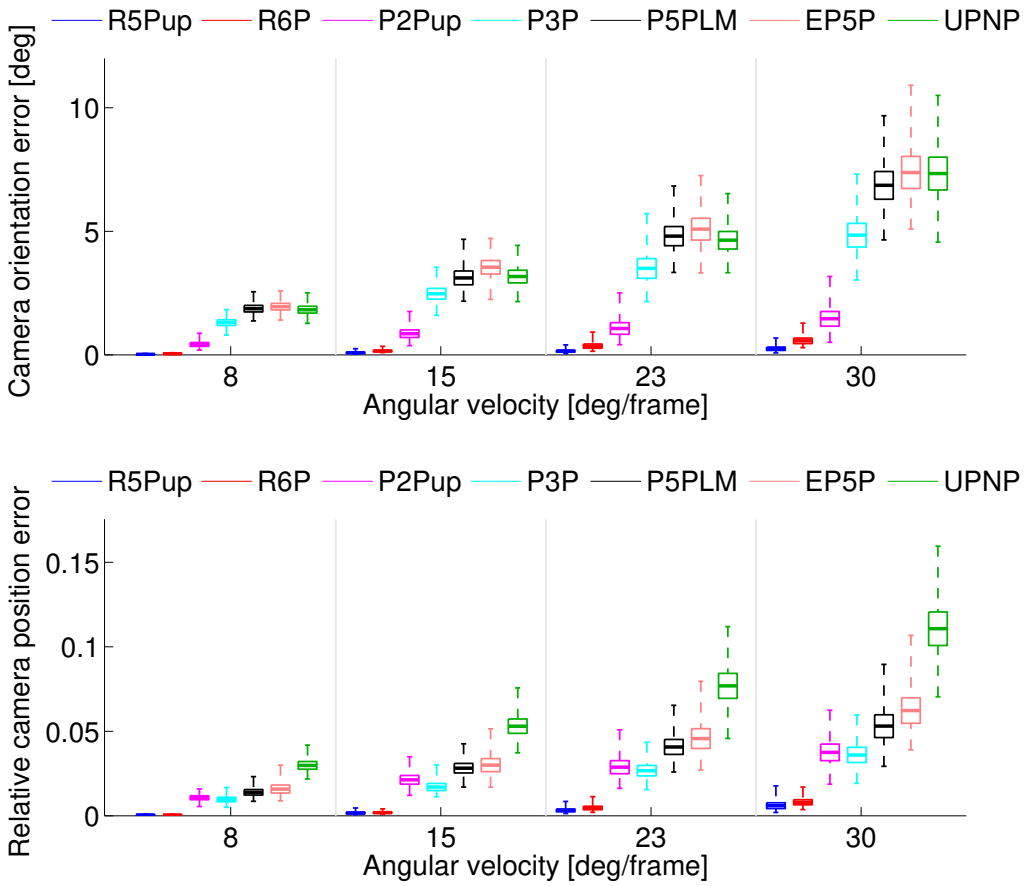


Figure 7.3: Comparing R5Pup pose estimates to other methods on data with varying camera angular velocity and translational velocity.

### 7.3.2 Real data

We focused on two typical use cases of absolute pose algorithms in our real experiments. The camera pose estimation for augmented reality and 3D model reconstruction using Structure from motion. Data was collected using a cellphone Samsung Galaxy S5 which recorded both images and the IMU measurements to provide upvectors.

For the first case, camera pose was estimated from data obtained by augmented reality library ArUco [93]. A planar marker was detected in the image providing twelve 2D-3D correspondences. Such marker can be used to set-up a coordinate system and place objects in the scene as in figure 7.6 From these twelve correspondences, five were chosen for camera pose computation using R5Pup, P5PLM and EP5P. Outer points were selected primarily in order to cover the most of the image area. For P3P and P2Pup three and two correspondences were selected respectively. In order to make the comparison fair, all possible pairs and triplets from the five points used by

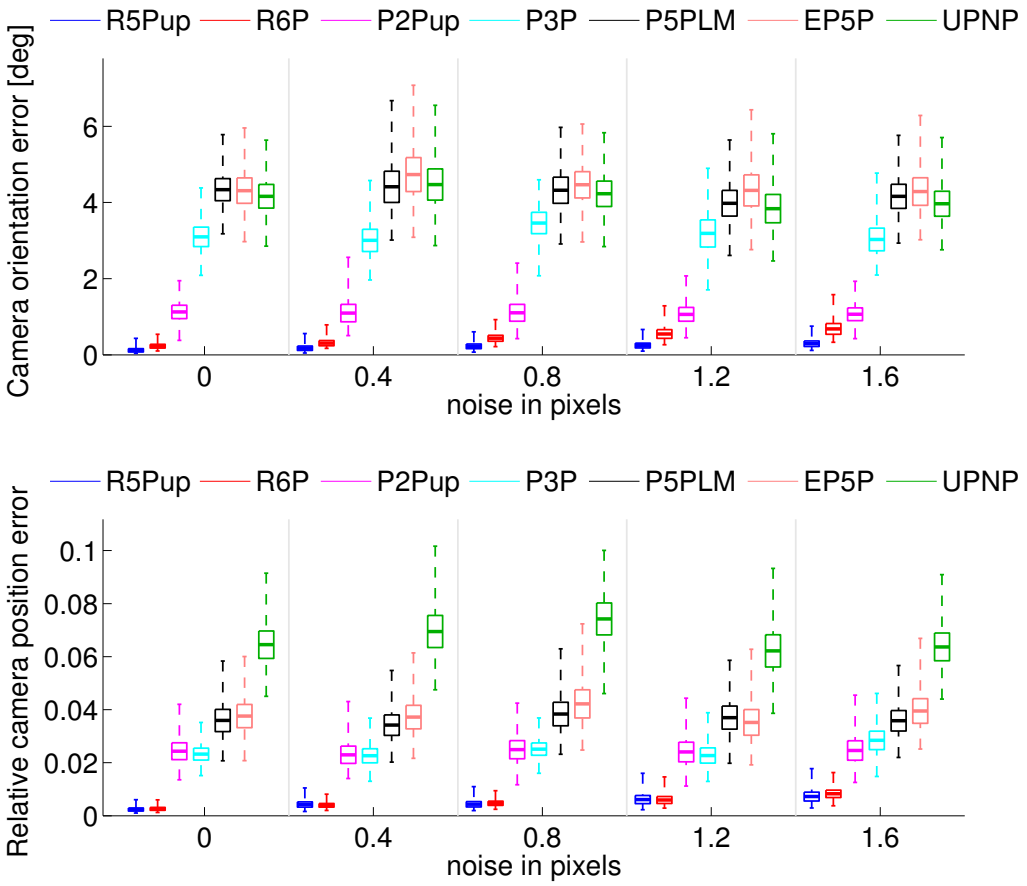


Figure 7.4: Comparing R5Pup pose estimates to other methods on data with varying noise in the 2D measurements.

other algorithms were tested. R6P was not evaluated here, since we found that it does not work on planar scenes.

Experiments focused on different camera motions to observe and identify cases where R5Pup brings improvement over standard algorithms. Five experiments were conducted with camera rotating in either roll, pitch or yaw and translating in x or y image direction. Each motion creates different RS distortion effects.

Results in figure 7.8 show that R5Pup models the distortions caused by moving RS camera better than all other methods. It is clear that some motions induce more difficult distortions for perspective camera models to handle than the others. The most noticeable difference between perspective camera model and our model is during translation along the x image axis. As the rows are read out sequentially in the direction of y axis, this causes skew effect in the image. In contrast to that, translating in the y direction causes shrinking or inflating along the x direction in the image. From the rotational movements, most significant problems for P3P were caused by

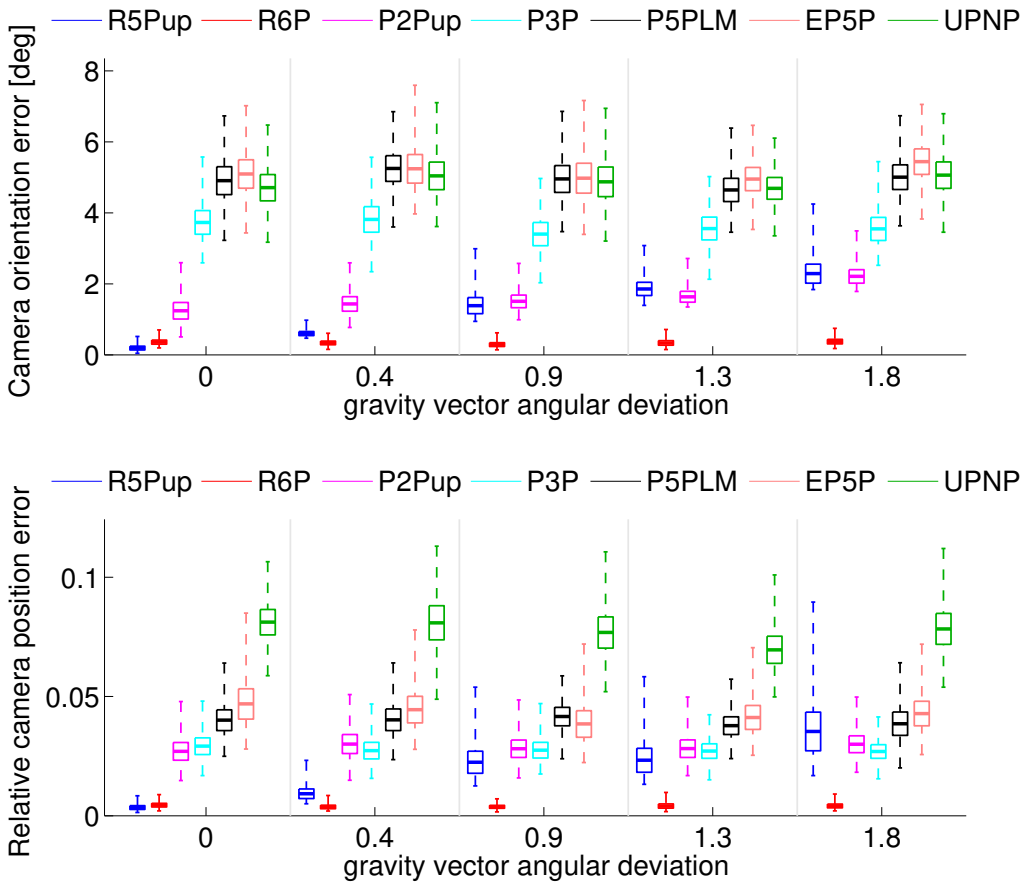


Figure 7.5: Introducing different errors on the vertical direction information.

yaw rotation, i.e., around the y axis in the image. This also causes skew effects, whereas pitch, the rotation around x image axis, causes again shrinking and extending in the y image axis.

Next experiment was aimed at determining the precision of the computed camera pose. In the absence of precise ground truth camera position data, we developed an experiment that shows the accuracy of retrieved camera poses. ArUco marker of the size of 1m was printed and placed on a ground plane. To induce the RS effect in the measurement, we rotated the camera around its three axes as in the first experiment, but this time with no translation. The camera sensor motion is negligible compared to the distance of the camera from the pattern (around 1.5m) and we can consider the camera having constant projection centre.

Therefore, reconstructed camera centers should be approximately in one spot, which is their mean. The histograms of distances from the mean was measured and is shown in figure 7.9. Lower distance from the mean camera center means better result. The standard deviations of the distances are shown in table 7.1. R5Pup outperforms all the other algorithms which don't account for RS effect.

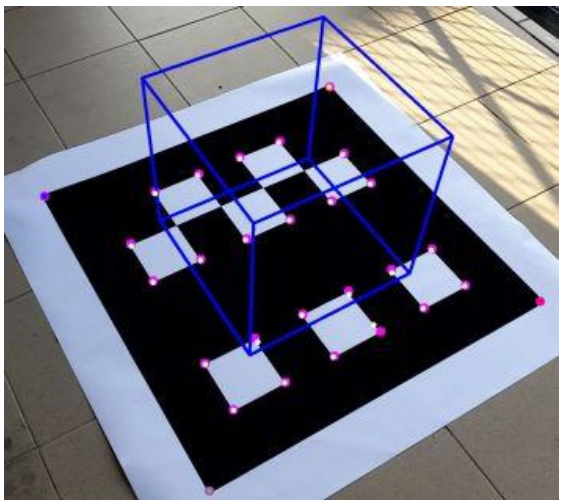

Figure 7.6: Using ArUco pattern to obtain 2D-3D correspondences. Camera pose estimation allows to place objects in the scene.

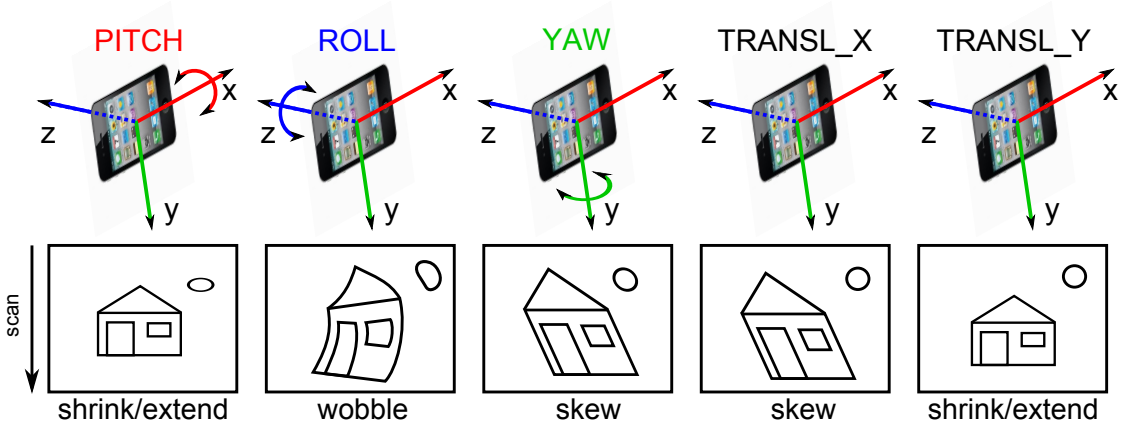


Figure 7.7: Visualization of the tested camera motions in real data experiments. Notice that the skew effect caused by translation along x axis as well as the shrink/extend effect caused by translation along y axis are different from the ones caused by yaw and pitch since they affect distant objects less and near objects more.

### Structure from Motion

A very interesting question is how will R5Pup perform when incorporated in a Structure from Motion pipeline working with real data. To investigate this, we developed a RS aware SfM pipeline which uses R5Pup to estimate absolute pose. A classic approach introduced in [103]

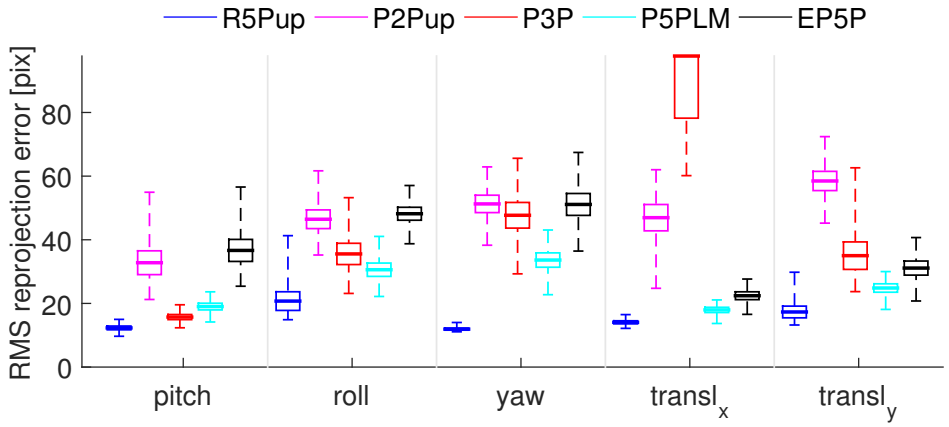


Figure 7.8: Mean reprojection error on the detected points of the ArUco pattern.

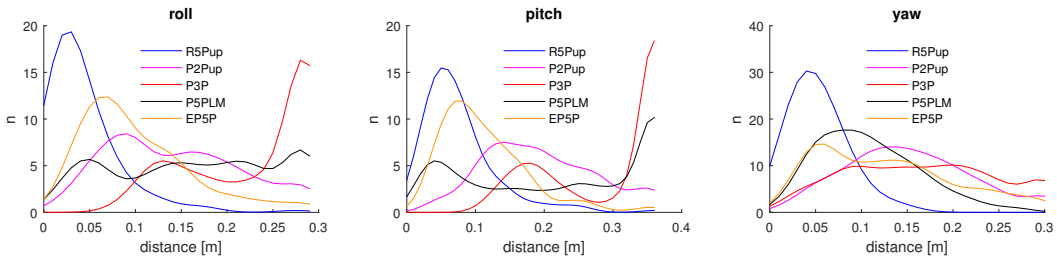


Figure 7.9: Camera center distances from the mean camera position. In the experiment, camera was purely rotating with no translation, therefore lower numbers mean better position estimation.

| | R5Pup | P2Pup | P3P | P5PLM | EP5P |
|---|---|---|---|---|---|
| roll | 0.325 | 0.527 | 0.868 | 0.543 | 0.346 |
| pitch | 0.128 | 0.379 | 0.822 | 0.329 | 0.286 |
| yaw | 0.111 | 0.590 | 0.343 | 0.200 | 0.269 |

Table 7.1: Histograms of distances from the mean camera center for the second ArUco experiment. In the experiment the camera center was not moving, therefore smaller distances mean better result.

was used and its key parts (point triangulation, bundle adjustment) were adjusted to incorporate the same RS model as in the R5Pup solver.

The initial geometry estimation is still global shutter, since there is no available RS relative pose algorithm. The initial cameras are, however, immediately optimized using BA with RS

model. After that, new cameras are added to the model using R5Pup and the RS parameters $\boldsymbol{\omega}$ and $\mathbf{t}$ are used throughout the optimization.

Due to transformations and deformations occurring during the reconstruction, the upvector direction in the scene is not guaranteed to remain $[0, 1, 0]$. An obvious solution would be to fix the upvector directions measured by the IMU so that the y-axis of each camera is fixed and only the rotation around y is optimized.

Unfortunately, we found the measurements from the cellphone IMU not precise enough for the reconstruction, which was poor or failed completely when the upvectors were fixed in the bundle adjustment.

To solve this issue, we developed the following approach. We don't force the upvectors to stay fixed during bundle adjustment. The upvectors are used only for adding new cameras using R5Pup. As mentioned before, this does not guarantee that the orientation of the scene will remain such that the upvectors of cameras would point upwards. This eventually causes problems when adding a new camera and the reconstruction fails. We solve this by aligning the subset of points which are used to estimate the new camera's pose such that their downward direction is as close to $[0, 1, 0]$ as possible. To do this, we find all the cameras which see the points from such subset, take the average of their upvector direction in the world coordinate frame represented by vector $\mathbf{g_{avg}}$ and find a rotation $\mathtt{R}_{align}$ such that $\mathtt{R}_{align}\mathbf{g_{avg}} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^\top$ and apply this rotation to the subset of points used for R5Pup. After obtaining the new camera's orientation with respect to the aligned points $\mathtt{R}_{local}$ we can compute the actual camera orientation in the scene as $\mathtt{R}_{scene} = \mathtt{R}_{local}\mathtt{R}_{align}$.

With this approach we have been able to reconstruct the datasets using upvectors from the cellphone IMU.

We compared our RS pipeline (R5P) to the widely known SfM pipeline Visual SfM [114] (VSFM) created by Changchang Wu. Data was obtained again using Samsung Galaxy S5 cellphone. We show only datasets where there was a observable qualitative difference between both methods. For the other datasets, the results were visually comparable. Results as well as sample pictures from the datasets are shown in figure 7.10.

Pictures from datasets House, Park and Street were captured while walking while holding the phone. Although there was some hand shaking, their camera trajectories should resemble a smooth line. Camera in dataset Tree was translating vertically and in dataset Bench horizontally. Dataset Door has the largest RS effect since the camera was moving and rotating quite rapidly with no specific pattern.

In dataset House, there is a noticeable scatter in the cameras reconstructed by VSFM whereas R5P gives a straight line ax expected. A noticeably larger portion of the building is reconstructed using R5P. Reconstruction of dataset Park failed completely using VSFM but was reconstructed well using R5P. In dataset Tree R5P reconstructed all 22 cameras, whereas VSFM only 11. Notice also the missing tree.

Dataset Street was reconstructed quite well using both methods but the trajectory of R5P cameras is again more smooth and also the house walls are more consistent. In dataset Door VSFM performed significantly worse presumably due to the large RS effect. Only the door was reconstructed using VSFM where R5P reconstructed much larger part of the visible scene. In

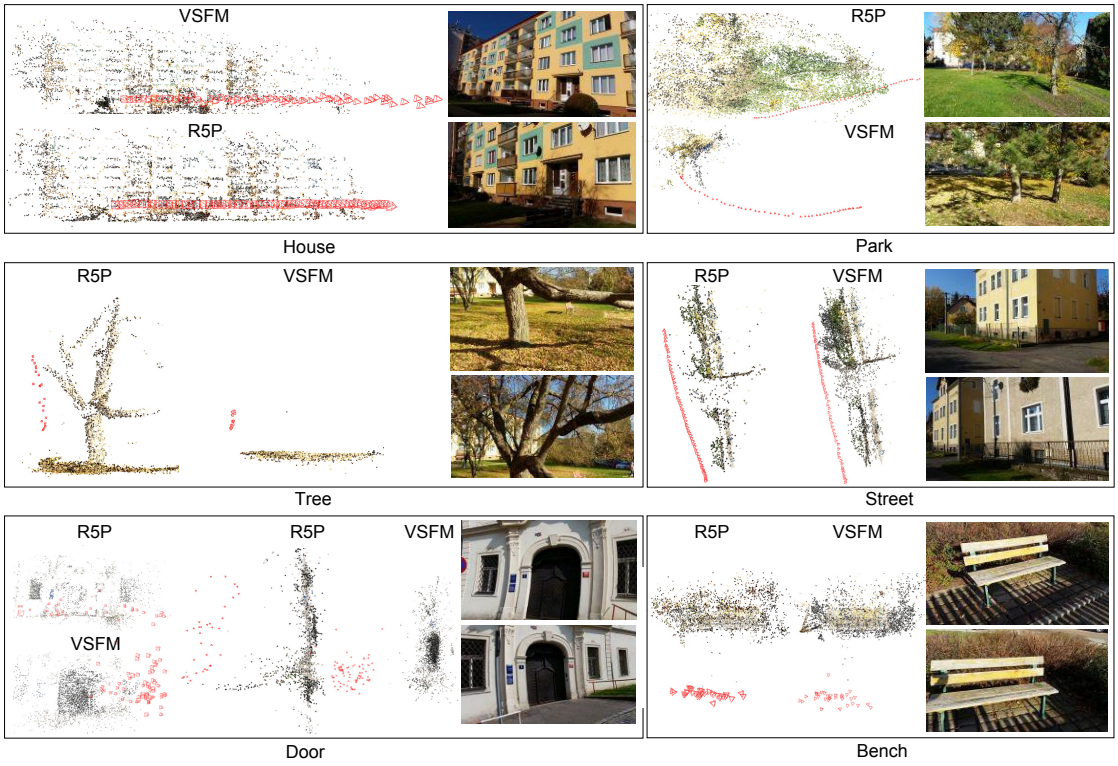


Figure 7.10: Structure from motion results on real data.

dataset Bench it is difficult to evaluate the quality of the model, but the cameras reconstructed by VSFM are significantly more scattered and some of them are off by a large amount.

## 7.4 Conclusion

In this chapter, we presented a solution to the rolling shutter absolute camera pose with known vertical direction. Compared to the general minimal solution R6P, knowing the vertical direction allows us to avoid double-linearization and to solve for the camera orientation directly without using P3P as an initialization. It also reduces the number of required 2D-to-3D correspondences to five. We have shown how to construct an efficient solver based on hidden variable resultant method. The solver gives up to 8 solutions and our implementation runs in 140 $\mu$s which is much faster than R6P [5]. We demonstrated the performance of the solver thoroughly on synthetic as well as real data. The synthetic experiments show great improvement in camera pose estimation precision on rolling shutter data. When the upvector is known precisely, the performance is at least the same or better than the performance of R6P. According to our experiments, we can expect improvements in camera pose estimation compare to the global shutter solvers up to the error of 1.5 degree in the vertical direction measurement. That is a value easily achievable using

high quality IMU sensors, but we have demonstrated that even using a common smartphone IMU we can obtain precise enough vertical direction measurements for the solver to outperform others. Last but not least, we have developed a RS aware SfM pipeline using the new R5Pup solver to incrementally add cameras to the scene. We have presented an approach to handling imprecise vertical direction measurements in such pipeline which is necessary in order to get a good reconstruction. By comparing to the state-of-the-art SfM piepline Visual SfM we have demonstrated the strengths of the R5Pup solver and its practical use for 3D reconstruction.

# 8 Linear solutions to rolling shutter absolute pose problem

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 [3, 80], planar scene [2], video sequences [48, 47, 60], are very slow [80] and provide too many solutions [5].

If one requires a practical algorithm similar to P3P, but working on RS images, the closest method available is R6P shown in chapter 6. However, R6P still needs around $1.4ms$ to compute the camera pose, compared to around $3\mu s$ 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 64 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.

In this chapter we introduce a different approach to the R6P problem. We create iterative solvers that still require only minimal number of points, provide a single solution and are much faster. The unknown parameters are split into groups and the algorithms alternate between the estimation of

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 $10\mu s$, which is up to $170\times$ faster than R6P;
- a 9-point linear non-iterative solver that provides more accurate camera pose estimates than R6P in $20\mu s$;
- another three 6-point iterative solvers that alternate between different camera parameters.

All solvers are easy to implement and they return a single solution. We formulate the problem of RS camera absolute pose in Section 8.1. Derivations of all new solvers are in Section 8.2. Section 8.3 contains experiments verifying the feasibility of the proposed solvers and it compares them against P3P and R6P [5].

## 8.1 Problem formulation

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 $\mathbf{T}$ are therefore functions of the image row as described in chapter 5,equation (6.1).

To describe those functions and keep the problem simple we will work with the model used in [5] that is presented 5.7 and uses a linear approximation to the camera orientation $\mathtt{R}(\mathbf{v})$. This model has the form

$$\lambda_i \begin{bmatrix} c_i \\ r_i \\ 1 \end{bmatrix} = (\mathtt{I} + (r_i - r_p)[\boldsymbol{\omega}]_\times)\,(\mathtt{I} + [\mathbf{v}]_\times)\,\mathbf{X}_i + \mathbf{T} + (r_i - r_p)\mathbf{t}. \tag{8.1}$$

The drawback of the model (8.1) is that $\mathtt{R}_\mathbf{v}$ is often not really small and thus cannot be really linearized. Thus, the accuracy of the model is dependent on the initial orientation of the camera in the world frame. In chapter 6, we have shown that the standard P3P algorithm [34] 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 close to the identity.

We will next show how to simplify this model by linearizing equation (8.1) further and yet still obtaining a similar performance as the Gröbner basis R6P absolute pose solver presented in chapter 6 and published in [5].

## 8.2 Linear rolling shutter solvers

In this section, we present several linear iterative solvers to the minimal absolute pose rolling shutter problem. All these solvers start with the model (8.1) and they use six 2D-3D image point correspondences to estimate 12 unknowns $\mathbf{v}, \mathbf{T}, \boldsymbol{\omega}$, and $\mathbf{t}$. The proposed solvers differ in the way how the system (8.1) is linearized. Additionally we propose a linear non-iterative 9 point absolute pose rolling shutter solver.

### 8.2.1 R6P$^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T}}$ solver

The R6P$^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T}}$ solver is based on the idea of alternating between two linear solvers. The first R6P$_{\mathbf{v},\mathbf{T}}$ solver fixes the rolling shutter parameters $\boldsymbol{\omega}$ and $\mathbf{t}$ in (8.1) and estimates only the camera parameters $\mathbf{v}$ and $\mathbf{T}$. The second R6P$_{\boldsymbol{\omega},\mathbf{t}}$ solver fixes the camera parameters $\mathbf{v}$ and $\mathbf{T}$ and estimates only the rolling shutter parameters $\boldsymbol{\omega}$ and $\mathbf{t}$. 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 $\mathbf{v}$ and $\mathbf{T}$ can be estimated quite accurately.

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

The R6P$^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T}}$ solver performs reasonably well for small rolling shutter effects, however, as we will show in experiments, for larger RS effects, it may diverge and provide quite bad estimates. This is caused by the fact that once the camera parameters become imprecise, the rolling shutter parameters estimated by the R6P$_{\boldsymbol{\omega},\mathbf{t}}$ may be almost random.

### 8.2.2 R6P$^{\boldsymbol{\omega}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ solver

To avoid problems of the R6P$^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T}}$ solver, we introduce the R6P$^{\boldsymbol{\omega}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ solver. The R6P$^{\boldsymbol{\omega}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ solver alternates between two solvers, i.e. the linear R6P$_{\mathbf{v},\mathbf{T},\mathbf{t}}$ solver, which fixes only the rolling shutter rotation $\boldsymbol{\omega}$ and estimates $\mathbf{v}$, $\mathbf{T}$ and $\mathbf{t}$, and the R6P$_{\boldsymbol{\omega}}$ solver that estimates only the rolling shutter rotation $\boldsymbol{\omega}$ using the fixed $\mathbf{v}$, $\mathbf{T}$ and $\mathbf{t}$. The R6P$_{\mathbf{v},\mathbf{T},\mathbf{t}}$ solver solves 12 linear equations in 9 unknowns and the R6P$_{\boldsymbol{\omega}}$ solver solves 12 linear equations in 3 unknowns in the least square sense. Since the first R6P$_{\mathbf{v},\mathbf{T},\mathbf{t}}$ solver assumes unknown rolling shutter translation, the camera parameters are estimated with better precision than in the case of the R6P$_{\mathbf{v},\mathbf{T}}$ solver. Moreover, in many applications, e.g. cameras on a car, cameras often undergo only a translation motion, and therefore $\boldsymbol{\omega}$ is negligible. In such situations, the first iteration of the R6P$_{\mathbf{v},\mathbf{T},\mathbf{t}}$ solver already provides very precise estimates of the camera parameters.

The R6P$^{\boldsymbol{\omega}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ solver, again, alternates between the two linear solvers until the desired precision is obtained or a maximum number of iterations is reached.

### 8.2.3 R6P$^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ solver

We have observed that sometimes the rolling shutter rotation $\boldsymbol{\omega}$ can be compensated by a wrongly estimated rolling shutter translation $\mathbf{t}$ while the camera parameters $\mathbf{v}$ and $\mathbf{T}$ are quite precisely estimated by the R6P$_{\mathbf{v},\mathbf{T},\mathbf{t}}$ solver. Therefore, we develop another R6P$^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ solver. It first uses R6P$_{\mathbf{v},\mathbf{T},\mathbf{t}}$ solver, as the R6P$^{\boldsymbol{\omega}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ solver from Section 8.2.2, but then it uses only the estimated $\mathbf{v}$ and $\mathbf{T}$ and with these estimates computes both rolling shutter parameters $\boldsymbol{\omega}$ and $\mathbf{t}$ by using the linear R6P$_{\boldsymbol{\omega},\mathbf{t}}$ solver. In other words, the solver R6P$^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ re-estimates in the second step the rolling shutter translation together with the rolling shutter rotation.

### 8.2.4 R6P$^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ solver

Solver R6P$^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ estimates all unknown parameters $\mathbf{v}$, $\mathbf{T}$, $\boldsymbol{\omega}$ and $\mathbf{t}$ together in one step. To avoid non-linear equations in (8.1), the solver fixes $[\mathbf{v}]_\times$ that appears in the nonlinear term $[\boldsymbol{\omega}]_\times[\mathbf{v}]_\times$ in (8.1). Thus the solver solves equations

$$\lambda_i \begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix} = (\mathtt{I} + (r_i - r_0)[\boldsymbol{\omega}]_\times)\,\mathbf{X}_i + [\mathbf{v}]_\times \mathbf{X}_i + (r_i - r_0)[\boldsymbol{\omega}]_\times[\hat{\mathbf{v}}]_\times \mathbf{X}_i + \mathbf{T} + (r_i - r_0)\mathbf{t}, \quad (8.2)$$

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

In the first iteration $\hat{\mathbf{v}}$, is set to the zero vector and the term $(r_i - r_0)[\boldsymbol{\omega}]_\times[\hat{\mathbf{v}}]_\times \mathbf{X}_i$ in (8.2) 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 $\boldsymbol{\omega}$ during the capture is usually small. Therefore, the nonlinear term $[\boldsymbol{\omega}]_\times[\mathbf{v}]_\times$ is small, sometimes even negligible, and thus it can be considered to be zero in the first iteration.

In the remaining iterations we fix $\hat{\mathbf{v}}$ in the $(r_i - r_0)[\boldsymbol{\omega}]_\times[\hat{\mathbf{v}}]_\times\mathbf{X}_i$ term to be equal to the $\mathbf{v}_i$ estimated in the previous iteration of the $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ solver. Note that we fix only $\mathbf{v}$ that appears in the nonlinear term $[\boldsymbol{\omega}]_\times[\mathbf{v}]_\times$ and there is still another term with $\mathbf{v}$ in (8.2) from which a new $\mathbf{v}$ can be estimated. The $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ 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 $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ provides very precise estimates already after 1 iteration and after 5 iterations it has visually the same performance as the state-of-the-art R6P solver [5].

### 8.2.5 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 equation (8.2) can be rewritten as

$$\lambda_i \begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix} = (\mathtt{I} + [\mathbf{v}]_\times)\,\mathbf{X}_i + \mathbf{T} + (r_i - r_0)([\boldsymbol{\omega}]_\times(\mathtt{I} + [\hat{\mathbf{v}}]_\times)\mathbf{X}_i + \mathbf{t}). \tag{8.3}$$

We can substitute the term $[\boldsymbol{\omega}]_\times(\mathtt{I}+[\hat{\mathbf{v}}]_\times)$ in (8.3) with a $3\times3$ unknown matrix $\mathtt{R}_{\text{RS}}$. After eliminating the scalar values $\lambda_i$ by multiplying equation (8.3) from the left by the skew symmetric matrix

$$\begin{bmatrix} 0 & -1 & c_i \\ 1 & 0 & -r_i \\ -c_i & r_i & 0 \end{bmatrix}, \tag{8.4}$$

and without considering internal structure of the matrix $\mathtt{R}_{\text{RS}}$, we obtain three linear equations in 18 unknowns, i.e. $\mathbf{v}, \mathbf{T}, \mathbf{t}$, and 9 unknowns in $\mathtt{R}_{\text{RS}}$. 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 (8.1) was an approximation to the real rolling shutter camera model and therefore the formulation with a general $3 \times 3$ matrix $\mathtt{R}_{\text{RS}}$ is yet a different approximation to this model.

## 8.3 Experiments

We tested the proposed solvers on a variety of synthetic and real datasets and compared the results with the original R6P solver [5] as well as P3P. We followed the general pattern of experiments used in [5] 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

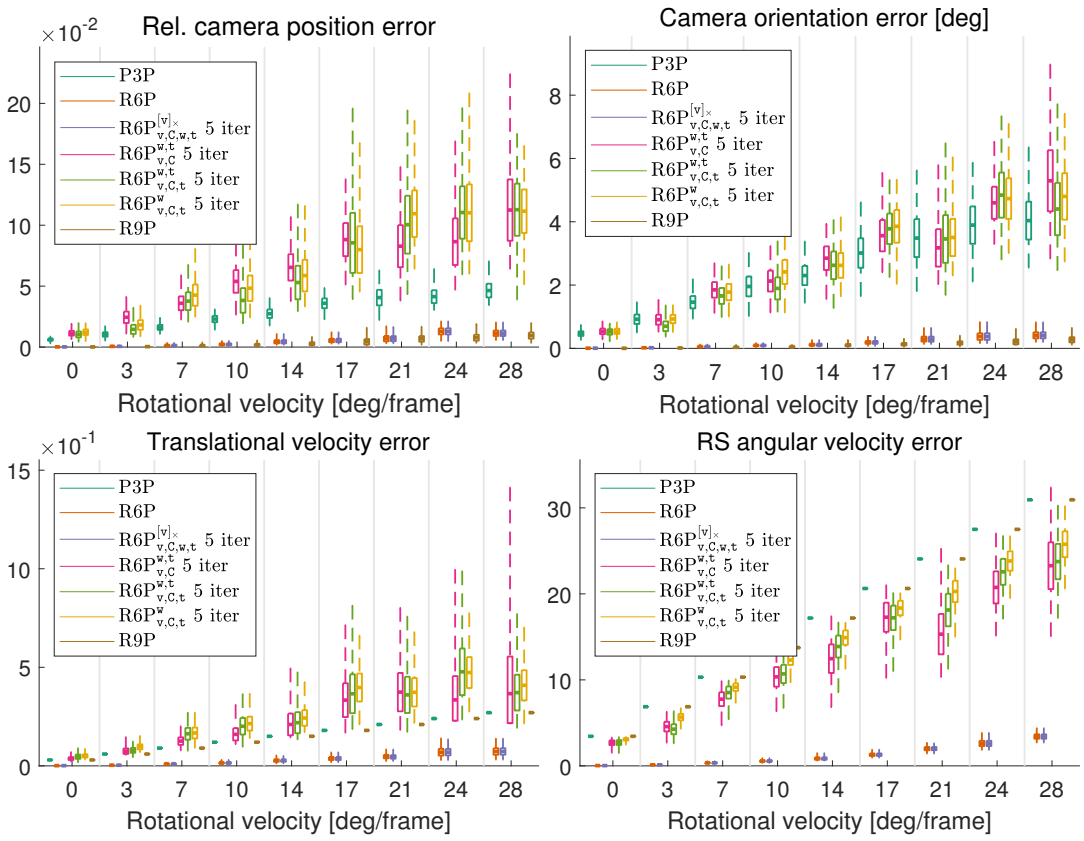


Figure 8.1: Experiment on synthetic data focusing on the precision of estimated camera poses and velocities. Notice that the performance of $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ is identical to R6P and both are slightly outperformed by R9P. Other linear solvers perform very poorly in all respects.

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 degrees. 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 they are not generated with the same model as the one that is estimated by the solvers (linear approximation to a rotation). Although they are all just models and we could have chosen another one, e.g. constant angular acceleration, we consider the constant angular velocity model is 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.

### 8.3.1 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 degrees per frame. Figure 8.1 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.

In agreement with [5], R6P provides much better results than P3P thanks to the RS camera model. The newly proposed solver $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ provides almost identical results to R6P at much lower computation cost (cf. Table 8.1). 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.

For $\text{R6P}^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T}}$, $\text{R6P}^{\boldsymbol{\omega}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ and $\text{R6P}^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$, the maximum 5 iterations might not be enough to converge to a good solution, whereas $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ seems to perform at its best. Therefore, we increased the maximum number of iterations. The results in Figure 8.2 show that the performance of $\text{R6P}^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T}}$, $\text{R6P}^{\boldsymbol{\omega}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ and $\text{R6P}^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ can be improved by increasing the maximum number of iterations to 50. However, it is still far below the performance of R6P, $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ and R9P.

Since all the proposed solvers have a linearized form of the camera orientation, in the same way as R6P [5], we tested how being further from the linearization point affects the performance (Fig. 8.3). 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 degrees per frame for the rotation. In [5] the we show that R6P outperforms P3P in terms of camera center estimation up to 6 degrees away from the initial $\mathtt{R}$ estimate and up to 15 degrees away from $\mathtt{R}$ for the camera orientation estimate. Our results in Figure 8.3 show similar behavior and identical results of R6P and $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$. R9P performs comparable to both, even slightly outperforming them in terms of camera orientation estimation.

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 use case for all R6P solvers. To make the figure more informative, we chose the number of iterations for $\text{R6P}^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T}}$, $\text{R6P}^{\boldsymbol{\omega}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ and $\text{R6P}^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ to be 50 as the 5 iterations already proved to be insufficient in the first experiment, see Figure 8.1. Also, we set the maximum number of iterations for $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ to 1, to demonstrate the potential of this solver.

As seen in Figure 8.4, $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ is able to provide 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 $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ is two orders of magnitude less than of R6P. With 50 iterations the other iterative solvers now perform better than P3P, but considering the computational cost of 50 iterations, which could be even higher than that of a R6P we cannot recommend using them in such a scenario.

The computation times for all the tested solvers are shown in Table 8.1. One iteration of $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ is two orders of magnitude faster than R6P. According to the experiments, even one

Table 8.1: Computation time per iteration for all used solvers. Average timings on 2.5GHz i7 CPU.

| solver | P3P | R6P | $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ | $\text{R6P}^{\boldsymbol{\omega}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ | $\text{R6P}^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ | $\text{R6P}^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T}}$ | R9P |
|---|---|---|---|---|---|---|---|
| time per iteration | $3\mu s$ | $1700\mu s$ | $10\mu s$ | $24\mu s$ | $30\mu s$ | $27\mu s$ | $20\mu s$ |
| max # of solutions | 4 | 20 | 1 | 1 | 1 | 1 | 1 |

iteration of $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ 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 $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ because it is non-iterative and runs only once and is therefore as fast as 2 iterations of $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$.

One iteration of $\text{R6P}^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T}}$, $\text{R6P}^{\boldsymbol{\omega}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ and $\text{R6P}^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ is around three times slower than $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ but still almost two orders of magnitude faster than R6P. The synthetic experiments show that a significant number of iterations is needed in order to improve the pose estimates over P3P. However, as seen in the next section, real experiments show that even 5 iterations can be enough to capture the same amount of inliers as R6P or even more and the augmented reality experiment shows that $\text{R6P}^{\boldsymbol{\omega}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ performs best in terms of reprojection errors among the tested solvers. The concrete application must therefore be considered when choosing the solver.

### 8.3.2 Real data

We used the publicly available datasets from [47] and we show the results of the same frames shown in [5] (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 [5] 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 [47] 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 Figure 8.5 show the number of inliers captures over the sequences of images. We see that the performance of $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ with 5 iterations is visually identical to R6P. The results of $\text{R6P}^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ and $\text{R6P}^{\boldsymbol{\omega}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ are also very similar and often outperform R6P and $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$, except for the most challenging images in the House dataset.

The performance of $\text{R6P}^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T}}$ 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 $\text{R6P}^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T}}$ being the only solver that does not estimate the translational velocity $\mathbf{t}$ 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 [93] markers in known positions. We set up the markers in such a way that they cover 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 200ms 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 $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ with 5 iterations take around 5ms (200fps) and $\text{R6P}^{\boldsymbol{\omega}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ takes around 12.5ms (80fps). We did not test solvers $\text{R6P}^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T}}$, $\text{R6P}^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ and R9P in this experiment. This is because the performance of $\text{R6P}^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T}}$ is unstable, the performance of $\text{R6P}^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ is almost identical to $\text{R6P}^{\boldsymbol{\omega}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ and with R9P we do not have a way to extract the camera motion parameters and reprojection without them does not provide fair comparison.

We evaluated the reprojection error in each frame on all the detected markers. The results are shown in Figure 8.6. All the rolling shutter solvers outperform P3P in terms of precision of the reprojections. $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ again provides identical performance to R6P. $\text{R6P}^{\boldsymbol{\omega}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ has a slight edge over the others, which is interesting, considering its poor performance on the synthetic data.

Figure 8.6 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.

## 8.4 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 [5].

The overall best performing $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ 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 and $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ is still approximately 34x faster. 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 ($\text{R6P}^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T}}$, $\text{R6P}^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$, $\text{R6P}^{\boldsymbol{\omega}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$) that alternate between estimating different camera pose and velocity parameters. These three solvers are slower than $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ but still almost two orders of magnitude faster than R6P. While not as precise as R6P or $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ 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 they do not offer the fastest and most precise results, they performed best in some of the experiments 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 $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ 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 chapter presents a big step forward in practical computation of rolling shutter camera absolute pose, making it more available in real world applications.

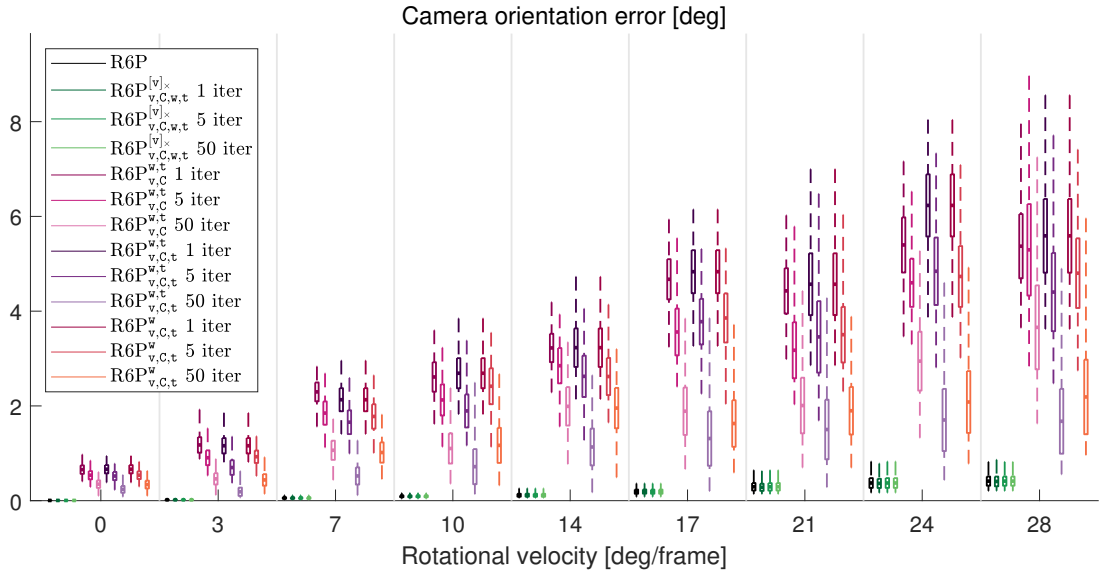


Figure 8.2: Testing the convergence of the iterative solvers. All iterative solvers have been run with 1, 5 and 50 iterations on data with $\mathtt{R} = \mathtt{I}$ and increasing RS effect exactly as in Figure 8.1. We see that increased number of iterations helps to improve the performance of R6P$^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T}}$, R6P$^{\boldsymbol{\omega}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ and R6P$^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$, but they still do not reach the precision of R6P and R6P$^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$. We can also see that R6P$^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ performs as well as the original R6P even with a single iteration, making it two orders of magnitude faster alternative.

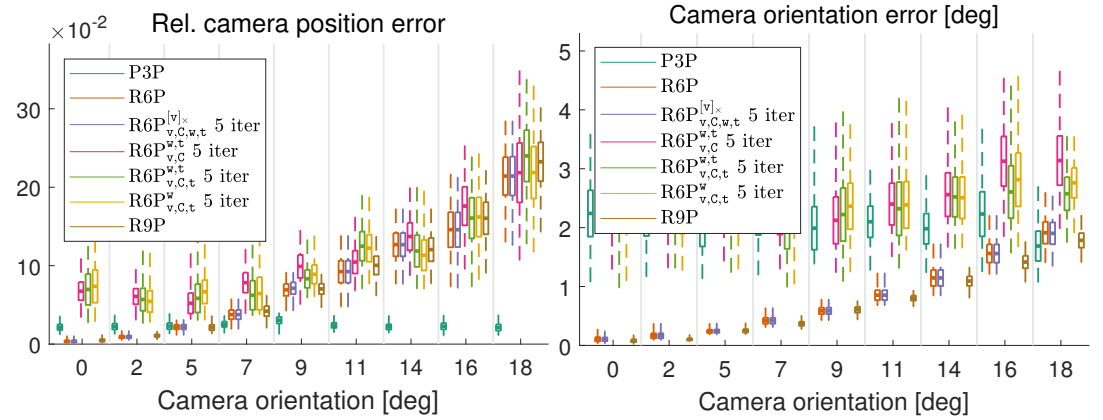


Figure 8.3: Experiment showing the effect of the linearized camera pose which is present in all models. The further the camera orientation is from the linearization point, the worse are the results. R6P$^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ matches the results of R6P and so does R9P.

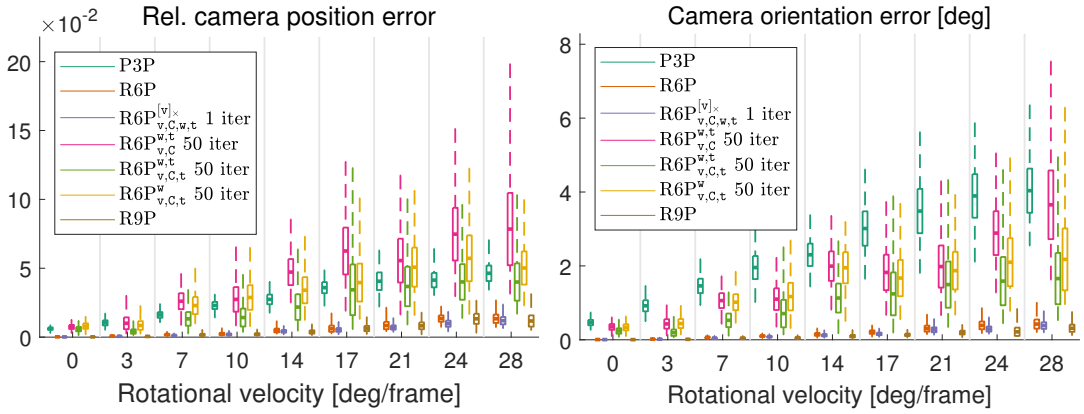


Figure 8.4: Increasing the camera motion and estimating camera pose with all solvers being initialized with P3P. $\text{R6P}^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ and R9P now provide consistently excellent results, comparable or outperforming those of R6P at a fraction of the computation cost. $\text{R6P}^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T}}$, $\text{R6P}^{\boldsymbol{\omega}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ and $\text{R6P}^{\boldsymbol{\omega},\mathbf{t}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ with 50 iterations now perform better than P3P, but still not as good as the other RS solvers.

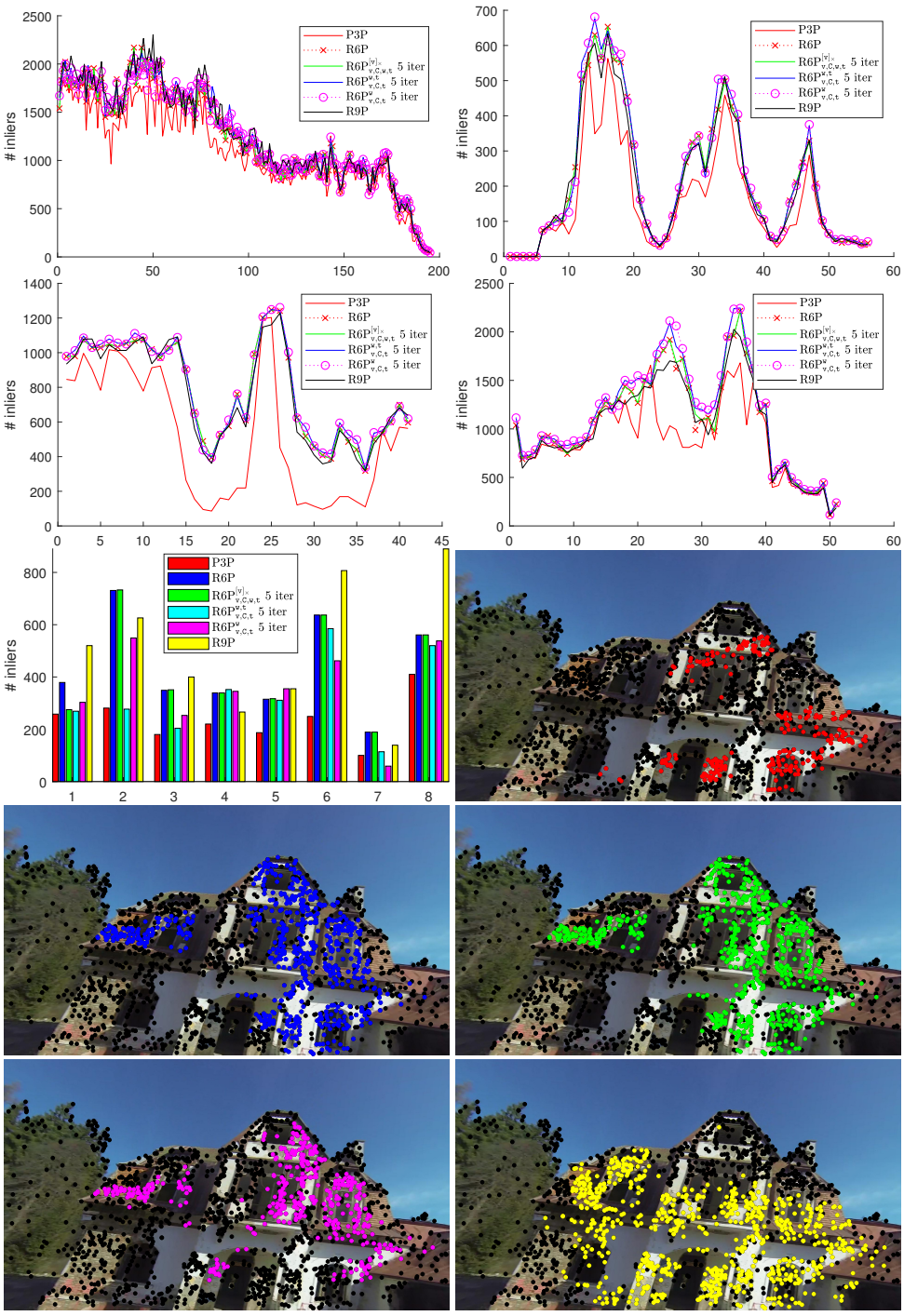


Figure 8.5: Number of inliers on real data sequences. From top to bottom, left to right: seq01, seq08, seq20, seq22 and House. The x axis contains frame numbers. The bar graph for the House figure is used because there is no temporal relationship between adjacent frames so a line graph does not make sense. Following are sample images from the House dataset frame 6, containing a high amount of RS distortion. The colored inliers in the sample images follow the same colors of algorithms as in the bar graph for House sequence.

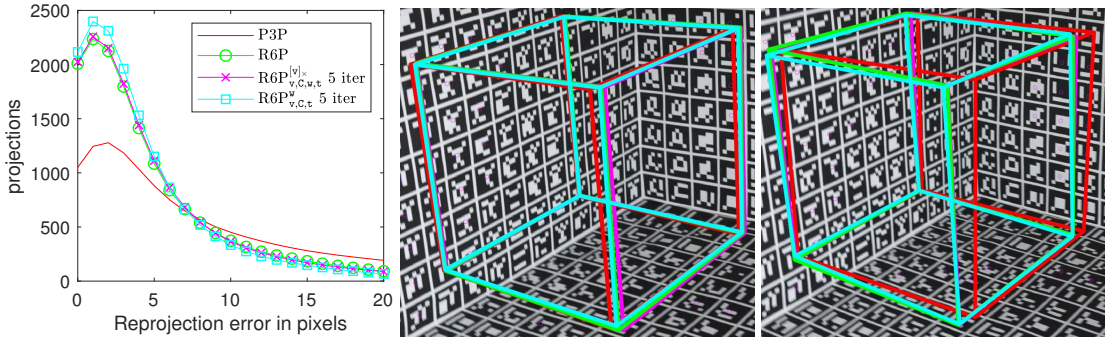


Figure 8.6: Histogram of reprojection errors on the Aruco markers in the augmented reality experiment. The rolling shutter absolute pose solvers (R6P in magenta,R6P$^{[\mathbf{v}]_\times}_{\mathbf{v},\mathbf{T},\boldsymbol{\omega},\mathbf{t}}$ in green, R6P$^{\boldsymbol{\omega}}_{\mathbf{v},\mathbf{T},\mathbf{t}}$ in cyan) keep the cube in place during camera motion whereas P3P (red) reprojects the cube all over the place.

# 9 Degeneracies in Rolling Shutter SfM

This chapter contains an analysis of degenerate configurations that arise when using rolling shutter camera models. The addition of new parameters and the ability to describe camera motion leads to more freedom in the scene geometry and allows us to explain images by degenerate geometry that is far from the reality.

## 9.1 Notation and concepts

A *similarity* transformation $\mathcal{S}$ is a composition of rotation $\mathtt{R}$, translation $\mathbf{T}$ and uniform scaling $s$, i.e. $\mathcal{S}(\mathbf{X}) = s\,\mathtt{R}\,\mathbf{X} + \mathbf{T}$, where $\mathtt{R}$ is a rotation matrix, $\mathbf{T}$ is a translation vector and $s$ is a scalar. *Image* $j$ is a set of vectors $\mathbf{u}_i^j \in \mathbb{R}^3 \setminus \{\mathbf{0}\}$ with $i = 1, \ldots, n$, $j = 1, \ldots, m$. *Scene* is a set of vectors $\mathbf{X}_i \in \mathbb{R}^3$. We consider only finite scenes for simplicity.

Scene points are projected to image points by cameras as $\lambda_i^j \mathbf{u}_i^j = \boldsymbol{\mu}(\mathtt{P}^j, \mathbf{X}_i)$, where $\mathtt{P}^j$ defines a particular camera projection model used and its parameters, and $\lambda_j^j$ are appropriate non-zero scales. For instance, when projecting by internally calibrated perspective cameras, the projection becomes $\lambda_i^j \mathbf{u}_i^j = \mathtt{R}^j \mathbf{X}_i + \mathbf{C}^j$, with rotation $\mathtt{R}^j$ and camera center $\mathbf{C}^j$.

A collection $\{\mathbf{X}_i, \mathtt{P}^j, \mathbf{u}_i^j\}$ such that $\lambda_i^j \mathbf{u}_i^j = \boldsymbol{\mu}(\mathtt{P}^j, \mathbf{X}_i)$ for some $\lambda_i^j$ is called a *configuration*. We say that configuration $\{\mathbf{X}_i, \mathtt{P}^j, \mathbf{u}_i^j\}$ *explains* images $\mathbf{u}_i^j$. We say that configuration $\{\mathbf{X}_i, \mathtt{P}^j, \mathbf{u}_i^j\}$ is *related* to configuration $\{\mathbf{Y}_i, \mathtt{Q}^j, \mathbf{u}_i^j\}$ by a similarity transformation when there is a similarity transformation of points $\mathbf{Y}_i = \mathcal{S}(\mathbf{X}_i)$ and camera projection models $\mathtt{Q}^j = \mathcal{S}(\mathtt{P}^j)$ such that $\beta_i^j \mathbf{u}_i^j = \boldsymbol{\mu}(\mathtt{Q}^j, \mathbf{Y}_i)$ for some $\beta_i^j$. For instance, for internally calibrated perspective cameras with $\mathtt{P}^j = (\mathtt{R}^j, \mathbf{C}^j)$, $\mathcal{S}(\mathtt{P}^j) = (\mathtt{R}^j\mathtt{R}^\top, s\mathbf{C}^j - \mathtt{R}^j\mathtt{R}^\top\mathbf{T})$ since $\boldsymbol{\mu}(\mathcal{S}(\mathtt{P}^j), \mathcal{S}(\mathbf{X_i})) = \mathtt{R}^j\mathtt{R}^\top(s\mathtt{R}\mathbf{X}_i + \mathbf{T}) + s\mathbf{C}^j - \mathtt{R}^j\mathtt{R}^\top\mathbf{T} = s\mathtt{R}^j\mathbf{X}_i + s\mathbf{C}^j = s\lambda_i^j\mathbf{u}_i^j$.

The goal of 3D reconstruction is to explain images $\mathbf{u}_i^j$ by a configuration $\{\mathbf{X}_i, \mathtt{P}^j, \mathbf{u}_i^j\}$ with scene points $\mathbf{X}_i$ measured in a Cartesian coordinate system. Different choices of Cartesian coordinate systems and different choices of measurement units produce configurations that are related by similarity transformations. Moreover, it is well-known that internally calibrated perspective images of a generic scene can be explained by a set $S$ of configurations that with every element $C$ of $S$ contains also all configurations related to $C$ by a similarity transformation [44], i.e. scene points can be reconstructed only up to a similarity transformation.

Therefore, every two configurations related by a similarity transformation will be considered equivalent. This equivalence relation partitions the set of all configurations into equivalence classes. Two configurations in one class are related by a similarity while two configurations in different classes are not related by a similarity. The equivalence class containing all config-

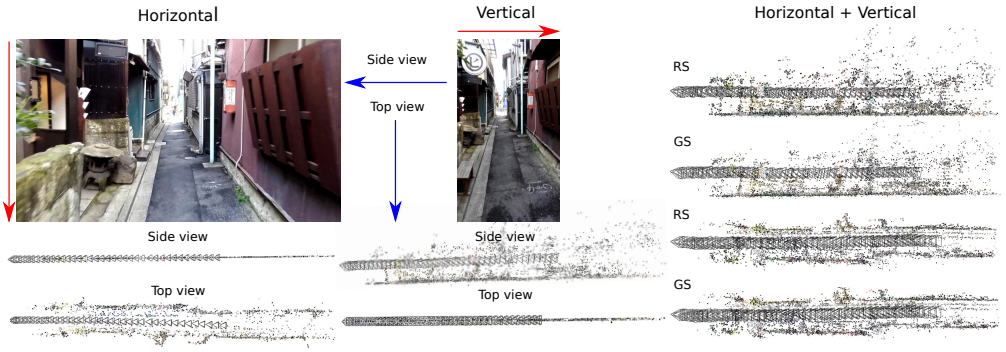


Figure 9.1: SfM with rolling shutter model can deliver undesired results. (Left) A reconstruction from forward camera translation with vertical readout direction. (Middle) A reconstruction of the same scene from forward moving camera horizontal readout direction. In both cases, the scene collapses into a plane that is perpendicular to the readout direction. (Right) When both image directions are combined a correct reconstruction is obtained with rolling shutter (RS) projection model, which is close to a reconstruction with global shutter (GS) model.

urations with scene points measured in a Cartesian coordinate system will be termed *correct reconstruction*. All other equivalence classes will be termed *incorrect reconstructions*.

We say that images $\mathbf{u}_i^j$ are *critical* if they can be explained by two configurations that are not equivalent, i.e., by at least one configuration that is in the incorrect reconstruction. Notice that our concept of criticality is somewhat different from concepts used in [43, 58], where they studied which scenes and cameras produce critical configurations for perspective images. Here we are interested in analyzing when images may be critical when using a rolling shutter models and we therefore modify the concept accordingly for that purpose.

## 9.2 Rolling shutter camera model

In this chapter, we consider internally calibrated rolling shutter (RS) camera models, which describe RS cameras that are realized as internally calibrated perspective cameras ($\mathtt{K} = \mathtt{I}$) with the row readout speed equal to one as described in chapter 5. To simplify the exposition, we will, hereafter, drop the adjective "internally calibrated". Therefore, "perspective model" means "internally calibrated perspective model" and "RS model" means "internally calibrated RS model".

Calibrated perspective projection can be described by $\lambda_i \mathbf{u}_i = \mathtt{R}\,\mathbf{X}_i + \mathbf{T}$ where $\mathtt{R} \in SO(3)$ and $\mathbf{C} \in \mathbb{R}^3$ is the rotation and the translation transforming a 3D point $\mathbf{X}_i \in \mathbb{R}^3$ from a world coordinate system to the camera coordinate system with $\mathbf{u}_i = [c_i, r_i, 1]^\top$, and $\lambda_i \in \mathbb{R} \setminus \{0\}$.

With RS camera, the right side of the equation (5.13) contains the image row $r_i$ as well. In general, there are two ways how to obtain the projection $\lambda_i \mathbf{u}_i = \lambda_i [c_i, r_i, 1]^\top = \mathtt{R}(r_i)\mathbf{X}_i + \mathbf{T}(r_i)$. We can treat $r_i$ on the right side as an unknown and then solve for both $r_i$ and $c_i$, which in general leads to multiple solutions, depending on the form of $\mathtt{R}(r_i)$ and $\mathbf{T}(r_i)$ we choose. The other option is to treat $r_i$ as known and make it equal to the measured row. The latter can be

justified by the measured row being the most precise measurement of the capture time we have. It was used in [47] for BA and proved to be adequate. We will therefore consider the image row that parameterizes $\mathtt{R}(r_i)$ and $\mathbf{T}(r_i)$ to be a known, constant parameter.

## 9.3 Bundle adjustment with independent RS models

In this chapter we consider Bundle Adjustment (BA) with independent RS models. This is more general than BA developed in [47] for (video) sequences of regularly spaced cameras where the camera motion during the image capture was constrained to be along the global camera trajectory. Our approach is necessary when reconstructing scenes from unorganized RS images.

We will base our derivations

Bundle adjustment [107] minimizes the sum of squares of reprojection errors which are, in our case, expressed as

$$\mathbf{e}_i^j = \tilde{\mathbf{u}}_i^j - \boldsymbol{\mu}(\mathtt{P}^j(\tilde{r}_i) * \mathbf{X}_i), \tag{9.1}$$

where $\tilde{\mathbf{u}}_i^j = \left[\tilde{c}_i^j, \tilde{r}_i^j\right]^\top$ is the measured image point and $\mathtt{P}^j(\tilde{r}_i)$ is an RS projection matrix of the $j$-th camera.

Non-linear least squares methods are used to find a solution $(\mathtt{P}^{j*}, \mathbf{X}_i^*)$ that (locally) minimizes the error over all the visible projections $(i, j)$

$$(\mathtt{P}^{j*}, \mathbf{X}_i^*) = \arg\min \sum_{(i,j)} \|\mathbf{e}_i^j\|^2.$$

When the set of images $\mathbf{u}_i^j$ is critical, it might happen that the bundle adjustment algorithm finds a local minimum producing an incorrect reconstruction. We will see that this indeed often happens.

## 9.4 Ambiguities in 3D reconstruction with RS camera models

Ambiguities in 3D reconstruction with the perspective projection model have been extensively studied in [58]. It has been found there that two perspective cameras and any number of scene points on certain ruled quadrics containing the cameras centers are in a critical configuration, as well as that for three perspective cameras, there is always a quartic curve of scene points such that they are in a critical configuration. Hence, there are situations when a set of perspective images become critical. However, the critical perspective images are very special and therefore do not in general pose problems for 3D reconstruction in practical situations with many points in generic scenes.

RS models are more general than the perspective model and therefore we expect to see more critical image sets when reconstructing with RS models. In particular, every perspective image can be explained by an RS model (5.21) with $\mathbf{t} = \mathbf{0}$ and $\mathtt{R}_r(r_i) = \mathtt{I}$. Therefore, every set of images that is critical for the perspective projection model is also critical for RS model (5.21).

RS cameras produce images with large variation of RS effects. Photographing static scenes with static RS cameras produces perspective images while images taken by RS cameras on a fast train exhibit pronounced RS effects. It is therefore desirable to look for RS SfM that can deal with all levels of RS effects. In particular, it is important that any practical RS SfM can handle perspective images.

When RS images are not truly perspective, it is often possible to treat the rolling shutter effect as (perhaps systematic) image error and explain RS images with perspective cameras, distorted scene, and somewhat higher image error. Therefore, it is important to analyze when a set of perspective images become critical w.r.t. RS model (5.21).

We will next show that, in many practical situations, images taken by perspective cameras become critical when reconstructed with RS model (5.21) and, even worse, when image noise is present, images can be explained by incorrect reconstructions with smaller error than is the smallest error of a correct reconstruction. Hence, in such situations, BA often prefers incorrect reconstructions.

We will use the RS camera model equation (5.21) with the rotation parameterized by the linearized model equation (5.21), which was used in [5, 2, 81, 80], since it is simple to show the ambiguities algebraically with this model. The linearized rotation model is an approximation to all the other models used in the literature and therefore images that are critical w.r.t. to model (5.21) will be close to critical for all other models if a cameras make turns by a small angle during the image capture.. For other models, the derivations we show will not hold exactly but they will be very close for many practical situations. We have observed in experiments that BA converges to incorrect reconstructions for all RS camera models in all the cases we have tested.

### 9.4.1 Single camera

We will start with showing how we can arbitrarily rotate the projection rays of a single RS camera and even collapse them in a single plane.

In order for a 3D point $\mathbf{X}_i$ to project into coordinate $[c_i, r_i, 1]^\top$ in the image, it has to lie on a plane defined by the row $r_i$ and the camera center. All points that lie in such a plane can be therefore described as

$$\mathbf{X}(c, r_i, \lambda) = (\mathtt{R}_r(r_i)\,\mathtt{R}_0)^{-1}\left(\lambda\,[c, r_i, 1]^\top - \mathbf{C}_0 - r_i\mathbf{t}\right).$$

To obtain an equation representing the plane, we need three non-collinear points, e.g.

$$\begin{aligned}
\mathbf{X}(1, r_i, 0) &= (\mathtt{R}_r(r_i)\,\mathtt{R}_0)^{-1}\left(-\mathbf{C}_0 - r_i\mathbf{t}\right), \\
\mathbf{X}(1, r_i, 1) &= (\mathtt{R}_r(r_i)\,\mathtt{R}_0)^{-1}\left([1, r_i, 1]^\top - \mathbf{C}_0 - r_i\mathbf{t}\right), \\
\mathbf{X}(0, r_i, 1) &= (\mathtt{R}_r(r_i)\,\mathtt{R}_0)^{-1}\left([0, r_i, 1]^\top - \mathbf{C}_0 - r_i\mathbf{t}\right).
\end{aligned}$$

The plane $\mathbf{n}(r_i)$ determined by these three points is the solution of the following homogeneous

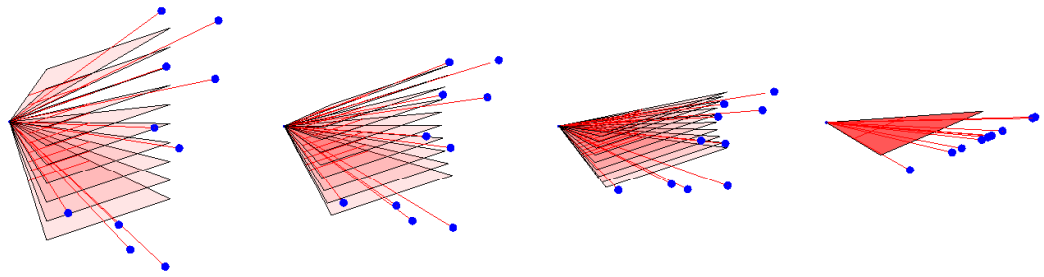

Figure 9.2: Changing the rotational velocity $\omega_x$ around the $x$-axis for a rolling shutter camera model changes the alignment of projection rays that correspond to each image row. From left to right there is $\omega_x = 0$, $\omega_x = -0.3$, $\omega_x = -0.6$ and $\omega_x = -1$. For $\omega_x = -1$ all projection rays collapse into a single plane and any image can be explained by 3D points in a plane.

equation system:

$$\begin{bmatrix} \left(-\mathbf{C}_0 - r_i\mathbf{t}\right)^\top \left(\mathtt{R}_r(r_i)\,\mathtt{R}_0\right)^{-\top} & 1 \\ \left([1, r_i, 1]^\top - \mathbf{C}_0 - r_i\mathbf{t}\right)^\top \left(\mathtt{R}_r(r_i)\,\mathtt{R}_0\right)^{-\top} & 1 \\ \left([0, r_i, 1]^\top - \mathbf{C}_0 - r_i\mathbf{t}\right)^\top \left(\mathtt{R}_r(r_i)\,\mathtt{R}_0\right)^{-\top} & 1 \end{bmatrix} \mathbf{n}(r_i) = \mathtt{A}(r_i)\mathbf{n}(r_i) = \mathbf{0} \qquad (9.2)$$

The solution of this system always spans at least one dimensional space, which is the null-space of $\mathtt{A}(r_i)$, since the rank of $\mathtt{A}(r_i)$ is at most three.

We set $\mathbf{C}_0 = [0, 0, 0]^\top$ and $\mathtt{R}_0 = \mathtt{I}$ for simplicity and disregard the translational motion $\mathbf{t}$. We then set $\omega_y = \omega_z = 0$ to simulate the rotation around the $x$-axis alone. The 3D point projected on a row $r_i$ is now written as $\mathbf{X}(c, r_i, \lambda) = \lambda \mathtt{R}_r(r_i)^{-1}\,[c, r_i, 1]^\top$. We again choose the triplet $\mathbf{X}(1, r_i, 0)$, $\mathbf{X}(1, r_i, 1)$ and $\mathbf{X}(0, r_i, 1)$ to determine the plane $\mathbf{n}(r_i)$, from which Eq. equation (9.2) yields

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & \frac{r_i(\omega_x+1)}{r_i^2\omega_x^2+1} & -\frac{r_i^2\omega_x-1}{r_i^2\omega_x^2+1} & 1 \end{bmatrix} \mathbf{n}(r_i) = 0.$$

We see that $\mathbf{n}(r_i)$ below is a solution:

$$\mathbf{n}(r_i) = \left[0, 1, \frac{r_i\ (\omega_x + 1)}{r_i^2\omega_x - 1}, 0\right]^\top.$$

We can see that if we set $\omega_x = -1$ then $\mathbf{n}(r_i)$ becomes the plane $y = 0$ for any $r_i$. This indicates that there exists a rotational motion (linearized) making all the projected planes $\mathbf{n}(r_i)$ coplanar (see Fig. 9.2).

We will now extend this example to a camera whose center lies in a plane $y = 0$ and whose corresponding $\mathbf{n}(0)$ is also contained in this plane. Such a camera has $\mathbf{C} = [C_x, 0, C_z]^\top$ and

can be rotated around the $y$-axis by angle $\phi$. We will now consider the translational motion $\mathbf{t} = [t_x, t_y, t_z]^\top$ as well. The null-space of the matrix $\mathtt{A}(r_i)$ then changes to

$$\left[-\sin(\phi)(\omega_x + 1), \frac{\omega_x r_i^2 - 1}{r_i}, \cos(\phi)(\omega_x + 1), C_z - t_y + r_i t_z\right]^\top.$$

It is clear that by setting $\omega_x = -1$, $t_z = 0$ and $t_y = C_z$, we obtain again the plane $y = 0$ for any $r_i$. We remark that we need the non-zero $t_y$ which is dependent on the camera position in the plane. The reason for this is that in this camera model we express the camera center in the camera coordinate system, which is changing for each $r_i$ due to $\omega_x$. We show that for the linearized rotation model the rotational velocity $\boldsymbol{\omega} = [\omega_x, 0, 0]^\top$ can be compensated by translational velocity $\mathbf{t} = [0, C_z, 0]^\top$ to fix the camera center in the world coordinate system.

### 9.4.2 Two cameras

Using the findings in the previous section, that arbitrary RS camera can be collapsed in a plane, we will now argue that every two images can be explained by two RS cameras and a planar scene such that the reprojection error equation (9.1) is zero.

Since each image can be explained by a camera whose center lies in plane $y = 0$ and this plane also contains all their projection rays, every two rays must intersect at least in one point. We can show this algebraically by using the equations for triangulating 3D points with known camera parameters. We can write the projection matrix parameterized by $r_i$ as

$$\mathtt{P}^j(r_i) = \begin{bmatrix} \mathtt{R}^j(r_i^j)\mathtt{R}_0^j & \mathbf{C}_0^j + \mathbf{C}_r^j(r_i^j) \end{bmatrix} = \left[\mathbf{p}_1^j(r_i^j), \mathbf{p}_2^j(r_i^j), \mathbf{p}_3^j(r_i^j)\right]^\top.$$

Then for a 3D point corresponding to two image measurements $\tilde{\mathbf{u}}_1 = \left[\tilde{c}_i^1, \tilde{r}_i^1\right]^\top$ and $\tilde{\mathbf{u}}_2 = \left[\tilde{c}_i^2, \tilde{r}_i^2\right]^\top$ in two cameras having parameters $\mathtt{P}^1(\tilde{r}_i^1)$ and $\mathtt{P}^2(\tilde{r}_i^2)$ the following system of equations must hold with $\lambda (\in \mathbb{R} \setminus \{0\})$

$$\mathtt{M}_i \mathbf{X}_i = \begin{bmatrix} \tilde{c}_i^1 \mathbf{p}_3^1(\tilde{r}_i^1)^\top - \mathbf{p}_1^1(\tilde{r}_i^1)^\top \\ \tilde{r}_i^1 \mathbf{p}_3^1(\tilde{r}_i^1)^\top - \mathbf{p}_2^1(\tilde{r}_i^1)^\top \\ \tilde{c}_i^2 \mathbf{p}_3^2(\tilde{r}_i^2)^\top - \mathbf{p}_1^2(\tilde{r}_i^2)^\top \\ \tilde{r}_i^2 \mathbf{p}_3^2(\tilde{r}_i^2)^\top - \mathbf{p}_2^2(\tilde{r}_i^2)^\top \end{bmatrix} \begin{bmatrix} \lambda x_i \\ \lambda y_i \\ \lambda z_i \\ \lambda \end{bmatrix} = \mathbf{0}. \tag{9.3}$$

In order for a 3D point $[x_i, y_i, z_i]^\top$ to exist, the null-space of the 4x4 matrix $\mathtt{M}_i$ has to be at least one-dimensional, i.e., the rank must be at most 3. To calculate the triangulated 3D point coordinates we can compute the null-space. For perspective cameras in general configuration, the null-space will be either zero dimensional for non-intersecting camera rays or one dimensional, corresponding to a single 3D point.

Let us apply Eq. equation (9.3) to the above example with two RS cameras whose centers both lie in a plane $y = 0$. The rotation matrices $\mathtt{R}_0^1$ and $\mathtt{R}_0^2$ will be rotations around $y$ axis by angles $\phi^1$ and $\phi^2$. Camera centers will lie anywhere in $y = 0$: $\mathbf{C}_0^1 = \left[C_x^1, 0, C_z^1\right]^\top$ and $\mathbf{C}_0^2 = \left[C_x^2, 0, C_z^2\right]^\top$. To collapse the projection rays of both cameras we will set, as shown in

the previous section, the rotational velocities $\omega_x^1 = -1$ and $\omega_x^2 = -1$ and translational velocities $\mathbf{t}^1 = \left[0, t_y^1, 0\right]^\top$ and $\mathbf{t}^2 = \left[0, t_y^2, 0\right]^\top$. We then obtain the following matrix

$$\mathtt{M}_i = \begin{bmatrix} -\cos(\phi^1) - \tilde{c}_i^1 \sin(\phi^1) & -\tilde{c}_i^1 \tilde{r}_i^1 & \tilde{c}_i^1 \cos(\phi^1) - \sin(\phi^1) & C_z^1 \tilde{c}_i^1 - C_x^1 \\ 0 & -(\tilde{r}_i^1)^2 - 1 & 0 & 0 \\ -\cos(\phi^2) - \tilde{c}_i^2 \sin(\phi^2) & -\tilde{c}_i^2 \tilde{r}_i^2 & \tilde{c}_i^2 \cos(\phi^2) - \sin(\phi^2) & C_z^2 \tilde{c}_i^2 - C_x^2 \\ 0 & -(\tilde{r}_i^2)^2 - 1 & 0 & 0 \end{bmatrix}.$$

The rank of $\mathtt{M}_i$ is at most 3 and, therefore, the rays always intersect at least in one point. For any pair of image projections the null-space of $\mathtt{M}_i$ and thus the subspace where the 3D point can lie is $[a, 0, b, 1]^\top$ and therefore all points could be reconstructed in plane $y = 0$.

### 9.4.3 Projecting onto a plane

Before we proceed to analysis of multiple RS cameras we need to explain an important fact, that is, the rotation induced by $\boldsymbol{\omega} = [-1, 0, 0]^\top$ in the linearized RS model is actually a projection onto a plane $y = 0$. Let us see what happens to an arbitrary point on a camera ray. Any 3D point that lies on a camera ray can be expressed in the camera coordinate system as $\lambda \left[c_i, r_i, 1\right]^\top$. For the sake of simplicity we will consider $\mathtt{R}_0 = \mathtt{I}$ and $\mathbf{C}_0 = [0, 0, 0]^\top$ now. We can express the 3D point in a world coordinate system by

$$\mathbf{X} = \mathtt{R}(r_i)^{-1} \begin{bmatrix} \lambda c_i \\ \lambda r_i \\ \lambda \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{1+r_i^2} & \frac{-r_i}{1+r_i^2} \\ 0 & \frac{r_i}{1+r_i^2} & \frac{1}{1+r_i^2} \end{bmatrix} \begin{bmatrix} \lambda c_i \\ \lambda r_i \\ \lambda \end{bmatrix} = \begin{bmatrix} \lambda c_i \\ 0 \\ \lambda \end{bmatrix},$$

which shows that the $x$ and $z$ coordinates remain the same while the $y$ coordinate is dropped. An illustration of this is in figure 9.3.

### 9.4.4 Multiple cameras with parallel $y$ (readout) directions

We will now use the result from previous subsection to make the following statement.

**Theorem:** *Assume any number of images taken by perspective cameras with parallel y (readout) directions in space. Then, if there exists a reconstruction for such cameras using the perspective camera model, then there also exists a reconstruction using the RS camera model (5.21) with all cameras and 3D points lying in plane $y = 0$.*

This statement can be proven by combining the previous statements. The perspective reconstruction gives a set of 3D points $\mathbf{X}_i = [x_i, y_i, z_i]^\top$ and the cameras whose centers are $\mathbf{C}_0^j = \left[C_x^j, C_y^j, C_z^j\right]^\top$ and whose $y$ axes are aligned with the $y$ axis in the world coordinate system, where $j$ is the index for cameras and $i$ for 3D points. If we project the rays connecting $\mathbf{C}_0^j$ and $\mathbf{X}_i$ onto the plane $y = 0$ we will obtain the rays that pass through $\hat{\mathbf{C}}_0^j = \left[C_x^j, 0, C_z^j\right]^\top$ and

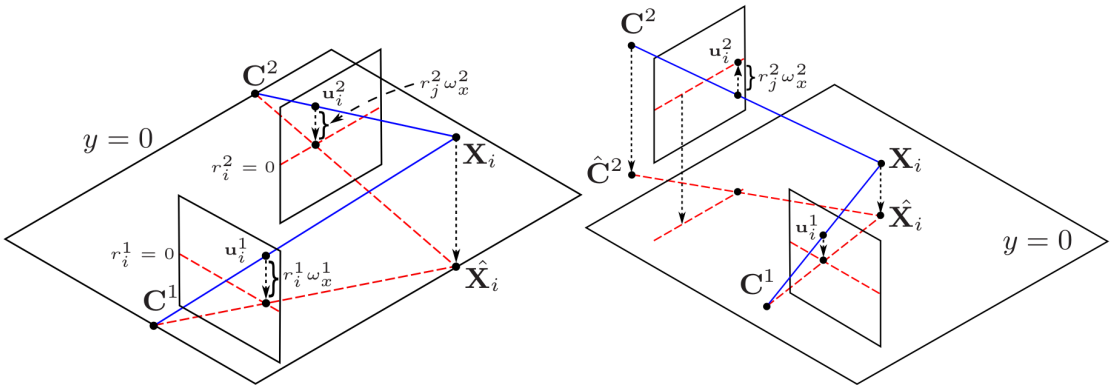


Figure 9.3: (Left) Two possible configurations of a scene from image projections $\mathbf{u}_i^1$ and $\mathbf{u}_i^2$. One is represented by two perspective cameras and point $\mathbf{X}_i$ and the other by linearized RS cameras with $\omega_x^1 = \omega_x^2 = -1$ and point $\hat{\mathbf{X}}_i$. This figure illustrates that changing $\omega_x$ parameter to $-1$ equals to a projection into a plane $y = 0$. (Right) This projection is possible even for cameras that do not lie in the plane $y = 0$ but their readout direction is parallel.

$\hat{\mathbf{X}}_i = [x_i, 0, z_i]^\top$ which we have shown that is a configuration that is easily achieved by setting $\omega_x^j = -1$ (see Fig. 9.3). It follows that if there exists a perspective reconstruction for such images with zero reprojection error, the reconstruction projected to $y = 0$ will also have a zero reprojection error.

### 9.4.5 The effect of planar projection in the presence of image noise

The mere existence of the planar representation of the scene is not a reason BA should converge to such a solution. In practice, however, measured image points are affected by noise, and this noise leads to non-zero reprojection error $\mathbf{e}_i^j$ in BA (see Eq. equation (9.1)). In this section we show that the planar projection always reduces the reprojection error and therefore it always provides a superior solution in BA.

Suppose measured image points $\tilde{\mathbf{u}}_i^j = \left[\tilde{c}_i^j, \tilde{r}_i^j\right]^\top$ are now affected by noise such that $\mathbf{e}_i^j = \tilde{\mathbf{u}}_i^j - \boldsymbol{\mu}(\mathtt{P}^j(\tilde{r}_i)\mathbf{X}_i)$. For perspective projection, i.e. $\boldsymbol{\omega}^j = [0,0,0]^\top$ and $\mathbf{t}^j = [0,0,0]^\top$ the error can be expressed as

$$\mathbf{e}_i^j = \tilde{\mathbf{u}}_i^j - \boldsymbol{\mu}\left(\mathtt{R}_0^j\mathbf{X}_i + \mathbf{C}^j\right) = \begin{bmatrix} \tilde{c}_i^j - \frac{C_x^j + x\cos(\phi^j) + z\sin(\phi^j)}{C_z^j + z\cos(\phi^j) - x\sin(\phi^j)} \\ \tilde{r}_i^j - \frac{y}{C_z^j + z\cos(\phi^j) - x\sin(\phi^j)} \end{bmatrix}$$

whereas the reprojection error using the linearized RS camera model with $\boldsymbol{\omega}^j = \left[\omega_x^j, 0, 0\right]^\top$ and

$\mathbf{t}^j = \left[0, C_z^j, 0\right]^\top$ is

$$\mathbf{e}_i^j = \begin{bmatrix} e_{ix}^j \\ e_{iy}^j \end{bmatrix} = \tilde{\mathbf{u}}_i^j - \boldsymbol{\mu}\left(\mathtt{R}_r^j(\tilde{r}_i^j)\mathtt{R}_0^j\mathbf{X}_i + \mathbf{C}^j + \tilde{r}_i^j\mathbf{t}^j\right) = \begin{bmatrix} \tilde{c}_i^j - \frac{C_x^j + x\cos(\phi^j) + z\sin(\phi^j)}{C_z^j + z\cos(\phi^j) - x\sin(\phi^j)} \\ 0 \end{bmatrix}.$$

The $e_{iy}^j$ component of the reprojection error is eliminated and the $e_{ix}^j$ component remains unchanged by the projection to $y = 0$; therefore the overall error is reduced. This is always true for images taken by the perspective cameras with identical $y$ directions in space.

### 9.4.6 What does it mean in practice?

We have shown the reason why the planar projection reduces the reprojection error in the case where all images are captured by perspective cameras with identical $y$ direction in space. This case is in practice hardly achieved exactly, but we can often come very close to this scenario, for example when taking handheld pictures while walking or taking pictures with a camera mounted on a car.

When images are captured with the $y$ directions not parallel, we are still able to reduce $e_{iy}^j$ to zero, but at the cost of increasing the $e_{ix}^j$ component. It follows that BA will try to reduce $e_{iy}^j$ as far as the increase in $e_{ix}^j$ does not exceed the reduction in $e_{iy}^j$.

The amount of increase in $e_{ix}^j$ depends on camera poses when images are taken and it is complicated to analyze in general. We have, however, practically observed the following fact.

**Observation:** *For three or more images by perspective cameras with pairwisely different $y$ directions, the deformation of the scene by BA due to using the RS model is directly dependent on the angle between the $y$ axes.*

In synthetic experiments, we show that when the smallest angle between the three pairs of $y$ directions is at least 30 degrees, the reconstruction is recovered correctly. I real experiments, on the other hand, we show that capturing the scene with sufficient amount of images with two distinct $y$ directions that are perpendicular with each other, i.e. taking portrait as well as landscape images provides a correct reconstruction.

## 9.5 Experiments

### 9.5.1 Synthetic experiments

In Section 9.4 we have shown that images captured with parallel readout directions used in BA with linearized RS camera model can be explained by a planar scene and that this configuration has lower reprojection error. In synthetic experiments we verified this also for SLERP and Rodriguez parameterization.

Further investigation was aimed at the case when image readout directions were not parallel during capture. We studied the amount of minimal angular difference between the three readout

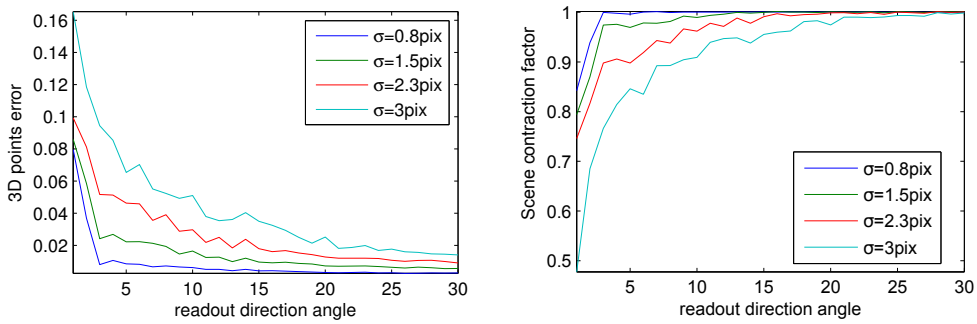


Figure 9.4: Experiment with three randomly initialized cameras. The $x$ axis shows the minimal readout direction angle among the three cameras. The figure on the left shows the mean spatial error over all 3D points after the optimization and the figure on the left shows the contraction factor of the scene compared to the ground truth. The lower the contraction factor the more deformation is in the scene. Results are shown for several values of error in the observations, expressed by the variance $\sigma$ of their zero mean normal distribution.

directions needed for the scene to be reconstructed correctly. To express the "correctness" of the reconstruction we introduce a measure which we call the scene contraction factor.

We calculate scene contraction factor as the ratio between the third principal component of the 3D points' coordinates before and after BA. The optimized 3D points after BA are first fitted to the initial 3D points by a similarity transform and then the principal components are calculated. If the scene is deformed, the third principal component will be different. A correctly reconstructed scene will have contraction factor close to 1 whereas completely flat scene will have contraction factor equal to 0.

We sampled three cameras randomly on a sphere with radius of 1 and pointing towards the a cubical scene. We measured the mean distance of initial 3D points from the resulting ones and also the scene contraction factor. Altogether 10,000 samples were generated and we categorized them based on the minimal angle between the three pairs of readout directions.

For each of these samples the same analysis as in previous experiment was done using 1000 different initializations with increasing image noise. We show the results for several values of the image noise in Fig. 9.4. From these experiments we can predict that if the minimal readout direction angle among the three camera pairs is at least 30 degrees, the reconstruction should be correct.

### 9.5.2 Real data experiments

To test our hypotheses under real conditions we captured several datasets using smart-phone camera under various angles. In order to have three mutually distinct RS readout directions we captured the same scene in vertical, horizontal and tilted position of the phone. Images were extracted from short videos captured handheld while moving around the objects or walking.

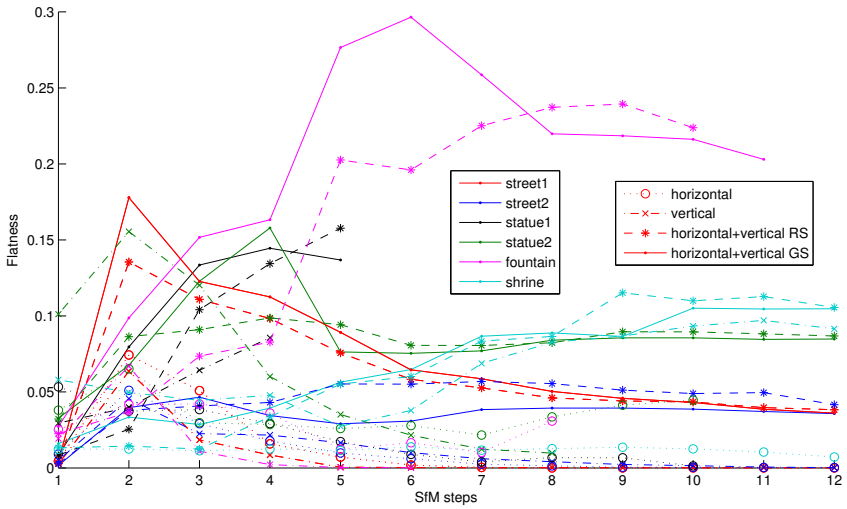


Figure 9.5: Analysis of the criticality in real datasets. The degeneracy is shown as flatness of the scene, where zero means completely flat. For either horizontal or vertical datasets, the degeneracy is apparent as the scene usually collapses to a plane completely. The results of the RS pipeline on datasets with both horizontal and vertical images show the same scene dimensions as the ones with GS pipeline.

An incremental SfM pipeline similar to [103, 114] was used to provide a baseline reconstruction. This pipeline was then adapted to use R6P [5] solver for absolute camera pose computation and used either linearized camera rotation model, SLERP or Rodriguez rotation model in bundle adjustment. Since we observed identical behavior for all rotation models, we present only the results of the linearized one. This rolling shutter aware version of the pipeline is denoted in the experiments as RS and the original as global shutter (GS) camera, which is equivalent with the perspective camera.

For each dataset we ran the two pipelines on several subsets of data – horizontal images, vertical images, horizontal+vertical and horizontal+vertical+tilted. According to our expectations, for the subsets containing only one readout direction the scene was collapsing to a plane as the RS incremental pipeline was progressing. We have calculated the flatness of the scene using principal component analysis (Fig. 9.5). Note that flatness in Fig. 9.5 is not the same as scene contraction factor used in synthetic experiments, value 0 still means the scene is completely flat, but the maximum is different for each dataset.

It is important to realize that in practice only few iterations of BA are allowed for the sake of performance and therefore the scene collapses gradually as new cameras are added and BA step is repeated. For small datasets with only small number of BA steps (up to 20 cameras) the deformation was not so apparent but it was extremely critical in larger datasets. When three distinct image readout directions were present in the dataset, we did not notice any deformation of the model caused by the RS pipeline compared to the GS pipeline, which confirms our predictions.

Even more important, however, are the experiments with two distinct readout directions (horizontal+vertical), which also do not show any deformation compared to the baseline GS reconstruction. This shows that in practice having horizontal as well as vertical images of the scene should be sufficient to successfully reconstruct the scene using RS pipeline. We show the results in (a rather complex) Fig. 9.6.

## 9.6 Conclusion

We tackled the topic of SfM with RS cameras. Recent works have shown that accounting for the camera movement in RS images can greatly improve the result and presented several practical RS camera models. We show that such models when used without constraints on the camera motion lead to incorrect reconstructions.

We analyzed the cases in which incorrect reconstruction arises and the reasons why it is so. We prove that any two perspective images can be explained by the linearized RS camera model and a planar scene. Further we prove that a set of images taken with parallel readout directions that can be explained by perspective cameras can also be explained by RS cameras and a scene all lying in a single plane. Moreover, we prove that the reprojection error is always reduced in such a case and, therefore, BA tends to prefer such solution.

This is a consequence of the linearized rotation being a mere projection on a plane. Since the linearized rotation model is a close approximation to all the other models it is expected that the other models will exert similar effects in BA. We have observed this both in synthetic and real data.

We show that in order to obtain a correct reconstruction using unconstrained RS SfM pipeline the input images should be captured with different readout directions. Synthetic experiments suggest that for 3 or more cameras, the minimal mutual angle between the readout directions should be at least 30 degrees. The experiments on real data confirm our predictions and in addition show that having two image sets with perpendicular readout directions is enough to obtain a correct reconstruction using SfM pipeline with the RS camera model.

Figure 9.6: Reconstructions using SfM pipeline for unorganized RS images. (Left) Horizontal image set sample and its reconstruction below. (Middle) Vertical image set sample and its reconstruction next to it. (Right) Reconstruction from both horizontal and vertical images. Notice the deformations when only one image direction is used. Two perpendicular directions provide correct results.

# 10 Two-View Geometry of Unsynchronized Cameras

This chapter presents a new method for *simultaneous computation of two-view camera geometry and temporal offset parameters from minimal sets of point correspondences*. We solve for fundamental matrix or homography together with temporal offset of image sequences. Our methods need only moving image point trajectories, which are easy to track. Unlike in [86, 84], we use a small (minimal) numbers of correspondences and we therefore are robust to outliers when combined with RANSAC robust estimation.

Secondly, we present an *iterative scheme using the minimal solvers to efficiently estimate large time offsets*. Our approach is based on RANSAC loop running our minimal solvers. This approach efficiently searches in the space of possible time offsets, which is much more efficient than exhaustive search methods [104, 106] developed before.

We evaluated our approach on a wide range of scenes and demonstrated its capability of synchronizing various kinds of real camera setups, such as driving cars, surveillance cameras, or sports match recordings with no other information than image data.

We demonstrate that our solvers are able to synchronize small time shifts of fractions of a second as well as large time shifts of tens of seconds. Our iterative algorithm is capable of synchronizing medium time shifts (i.e. tens of frames) with less than 5 RANSAC iterations and large time offsets (i.e. tens to hundreds of frames) using tens of RANSAC iterations. Overall, our approach is much more efficient than other methods utilizing RANSAC [87].

By solving two-camera synchronization problem, we also solve the multi-camera synchronization problem since temporal offsets of multiple cameras can be determined pairwise to serve as the initialization point for a global iterative solutions based on bundle adjustment [107].

## 10.1 Problem formulation

Let us consider two unsynchronized cameras with a fixed relative pose [44] producing a stereo video sequence by observing a dynamic scene. Motions of objects in the video sequence are indistinguishable from camera rig motions, and therefore, we will present the problem for static cameras and moving objects.

### 10.1.1 Geometry of two unsynchronized cameras

The coordinates of a 3D point moving along a smooth trajectory in space can be described by function

$$\mathbf{X}(t) = [X_1(t), X_2(t), X_3(t), 1]^\top, \tag{10.1}$$

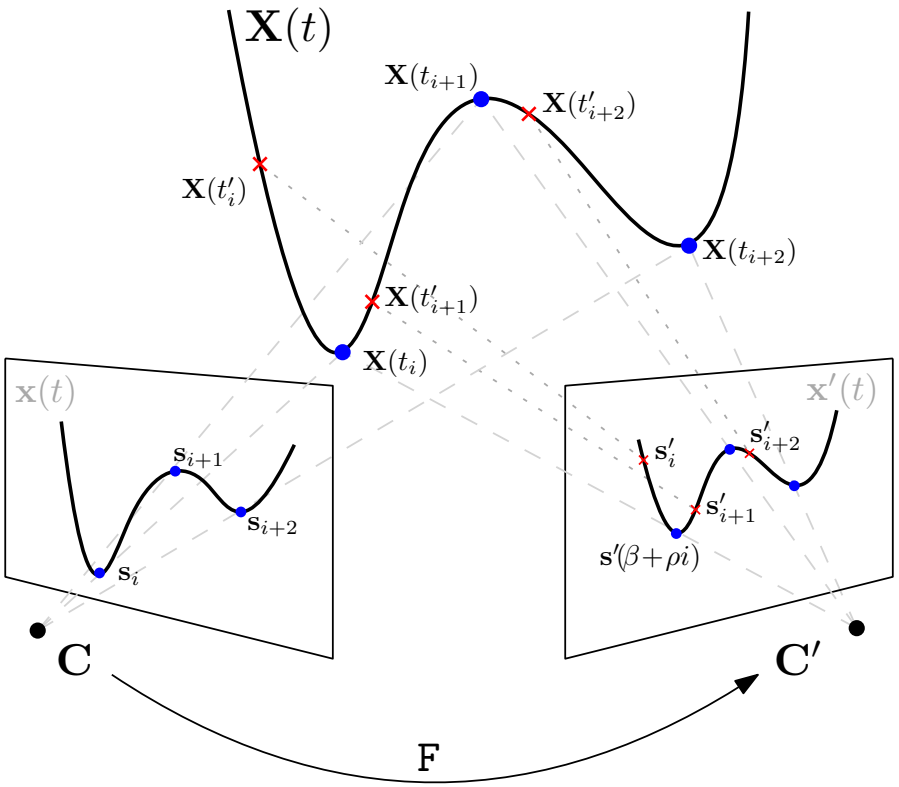


Figure 10.1: Two cameras capture a moving point at different times, so the projection rays of the two cameras meet nowhere.

where $t$ denotes time, see Figure 10.1. Projecting $\mathbf{X}(t)$ into the image planes of the two distinct cameras produces two 2D trajectories $\mathbf{x}(t)$ and $\mathbf{x}'(t)$. Now, let's assume that the first camera captures frames with frequency $f$ (period $p = 1/f$) starting at time $t_0$. This leads to a sequence of samples

$$\mathbf{s}_i = [u_i, v_i, 1]^\top = \mathbf{x}(t_i) = \boldsymbol{\pi}(\mathbf{X}(t_i)),\ i = 1, \ldots, n. \tag{10.2}$$

of the trajectory $\mathbf{x}(t)$ at times $t_i = t_0 + ip$.

Analogously, assuming a sampling frequency $f'$ (period $p' = 1/f'$), at times $t'_j = t'_0 + jp'$, the second camera produces a sequence of samples

$$\mathbf{s}'_j = [u'_j, v'_j, 1]^\top = \mathbf{x}'(t'_j) = \boldsymbol{\pi}'(\mathbf{X}(t'_j)),\ j = 1, \ldots, n'. \tag{10.3}$$

In general, there is no correspondence between the $s_i$ and $s'_j$ samples, *i.e.*, for $i = j$, $s_i$ and $s'_j$ do not represent the projections of the same 3D point. There are two main sources of desynchronization in video streams. The first one is the different recording starts or camera shutters triggering independently leading to a constant time shift. The second source are different

frame rates or imprecise clocks leading to different time scales. Assuming these two sources, we can map the time $t$ to $t'$ for frame $i$ using $\jmath(i) : \mathbb{N} \to \mathbb{R}$ as

$$\jmath(i) = \frac{t_i - t'_0}{p'} = \frac{t_0 + ip - t'_0}{p'} = \frac{t_0 - t'_0}{p'} + \frac{p}{p'}i = \beta + \rho i, \tag{10.4}$$

where $\beta \in \mathbb{R}$ is captures the time shift and $\rho \in \mathbb{R}$ the time scaling. Note that $\jmath(i)$ is an integer-to-real linear mapping with an analogous inverse mapping $\imath(j)$. Given the model in (10.4) and a sequence of image samples $\mathbf{s}'_j, j = 1, \ldots, n'$, we can interpolate a continuous curve $\mathbf{s}'(\jmath)$, for example using a spline, so that the 2D point corresponding to $\mathbf{s}_i$ is approximately given as

$$\mathbf{s}_i \longleftrightarrow \mathbf{s}'(\beta + \rho i). \tag{10.5}$$

Notice that the interpolated image curve $\mathbf{s}'(\cdot)$ is not equivalent to the true image trajectory $\mathbf{x}'(\cdot)$, but may be expected to be a good approximation under certain conditions. Even though it might appear reasonable to assume time shift to be known within a fraction of a second, it is often the case in practice that the timestamps are based on CPU clocks which together with startup delays can lead to time shift $\beta$ being in the order of seconds. On the other hand, the time scaling $\rho$ is more often known or can be calculated accurately.

### 10.1.2 Epipolar geometry

At any given (real-valued) time $t$, the epipolar constraint of the two cameras is determined by the following equation:

$$\mathbf{x}'(t)^\top \mathtt{F} \mathbf{x}(t) = 0. \tag{10.6}$$

For a sample $\mathbf{s}_i$ in the first camera, we can rewrite (10.6) using the corresponding point $\mathbf{x}'(t_i)$ in the second camera as

$$\mathbf{x}'(t_i)^\top \mathtt{F} \mathbf{s}_i = 0, \tag{10.7}$$

Using the approximation of the trajectory $\mathbf{x}'$ by $\mathbf{s}'$, we can express the approximate epipolar constraint as

$$\mathbf{s}'(\beta + \rho i)^\top \mathtt{F} \mathbf{s}_i = 0. \tag{10.8}$$

In principle, we can solve for the unknowns $\beta$, $\rho$, and $\mathtt{F}$ given 9 correspondences $s_i, s'_j$. However, such a solution would be necessarily iterative and too slow to be used as a RANSAC kernel. In the following, a further approximation is used to expresses the problem as a system of polynomials, which can be solved efficiently [65]. In section 10.5 we show an iterative solution built on this kernel, which can recover offsets of up to hundreds of frames.

### 10.1.3 Linearization of $\mathbf{s}'$ for known $\rho$

Let us assume that the relative framerate $\rho$ is known. In practice, the image curve $\mathbf{s}'$ is a complicated object. To arrive to our polynomial solution we approximate $\mathbf{s}'$ by the first order Taylor polynomial at $\beta_0 + \rho i$

$$\mathbf{s}'(\beta + \rho i) \approx \mathbf{s}'(\beta_0 + \rho i) + (\beta - \beta_0)\mathbf{v} = \mathbf{s}''(\beta + \rho i) \tag{10.9}$$

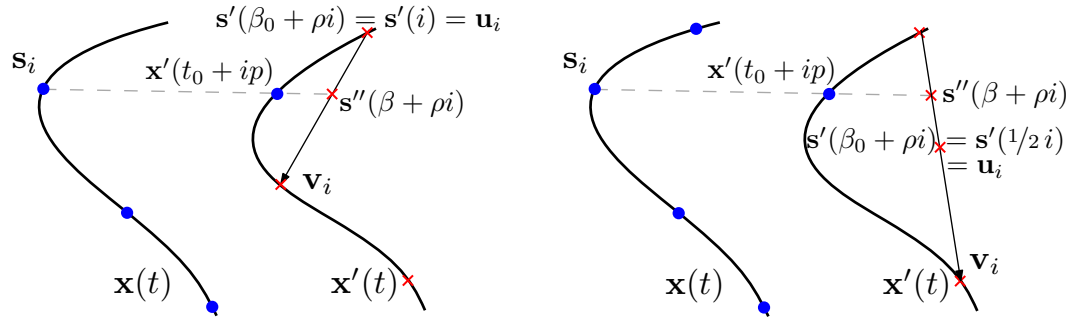


Figure 10.2: Illustration of the proposed trajectory linearization. (Left) Situation for $\rho = 1$, $\beta_0 = 0$ and $d = 1$ (Right) Situation for $\rho = 1/2$, $\beta_0 = 0$ and $d = 1$.

where $\mathbf{v}$ is the tangent vector $\dot{\mathbf{s}}'(\beta_0 + \rho i)$, and $\beta_0$ is an initial time shift estimate. We denote this approximation as $\mathbf{s}''$.

Further, we choose $\mathbf{v}$ to approximate the tangent over the next $d$ samples. Let $j_0 = \lfloor \beta_0 + \rho i \rfloor$ be the approximate discrete correspondence, and then

$$\mathbf{v} = \mathbf{s}'_{j_0+d} - \mathbf{s}'_{j_0}. \tag{10.10}$$

Note, that now $\mathbf{v}$ depends on $i$. For compactness, we write $\mathbf{u}_i = \mathbf{s}'(\beta_0 + \rho i) - \beta_0 \mathbf{v}_i$, and (10.8) becomes

$$(\mathbf{u}_i + \beta \mathbf{v}_i)^\top \mathtt{F} \mathbf{s}_i = 0 \tag{10.11}$$

In the rest of the chapter, we will assume that $f = f'$ and the initial estimate $\beta_0 = 0$. This situation is illustrated in Figure 10.2 (Left). However the key results hold for general known $\rho$, Figure 10.2 (Right), and $\beta_0 \neq 0$.

### 10.1.4 Homography

Using the same approach, we can write the equation for homography between two unsynchronized cameras. In the synchronized case, the homography between two cameras can be expressed as

$$\mathtt{H}\mathbf{s}_i = \lambda_i \mathbf{s}'_i. \tag{10.12}$$

where $\lambda_i$ is an unknown scalar. Approximating the image motion locally by a straight line gives for two unsynchronized cameras

$$\mathtt{H}\mathbf{s}_i = \lambda_i \left( \mathbf{u}_i + \beta \mathbf{v}_i \right). \tag{10.13}$$

## 10.2 Solving the equations

### 10.2.1 Minimal solution to epipolar geometry

The minimal solution to the simultaneous estimation of the epipolar geometry and the unknown time shift $\beta$ starts with the epipolar constraint equation (10.11). The fundamental matrix $\mathtt{F} =$

$[f_{ij}]_{i,j=1}^{3}$ is a $3 \times 3$ singular matrix, *i.e.* it satisfies

$$\det(\mathtt{F}) = 0. \tag{10.14}$$

Therefore, the minimal number of samples $\mathbf{s}_i$ and $\mathbf{s}'_i$ necessary to solve this problem is eight.

For eight samples in general position in two cameras, the epipolar constraint equation (10.11) can be rewritten as

$$\mathtt{M}\mathbf{w} = \mathbf{0}, \tag{10.15}$$

where $\mathtt{M}$ is a $8 \times 15$ coefficient matrix of rank 8 and $\mathbf{w}$ is a vector of monomials $\mathbf{w} = [f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33}, \beta f_{11}, \beta f_{12}, \beta f_{13}, \beta f_{21}, \beta f_{22}, \beta f_{23}]$. Since the fundamental matrix is only given up to scale, the monomial vector $\mathbf{w}$ can be parametrized using the 7-dimensional nullspace of the matrix $\mathtt{M}$ as

$$\mathbf{w} = \mathbf{n}_0 + \textstyle\sum_{i=1}^{6} \alpha_i \mathbf{n}_i, \tag{10.16}$$

where $\alpha_i$, $i = 1, \ldots, 6$ are new unknowns and $\mathbf{n}_i$, $i = 0, \ldots, 6$ are the null space vectors of the coefficient matrix $\mathtt{M}$. The elements of the monomial vector $\mathbf{w}$ satisfy

$$\beta w_j = w_k \quad \text{for } (j,k) \in \{(1,10), ..., (6,15)\}. \tag{10.17}$$

The parametrization equation (10.16) used in the rank constraint equation (10.14) and in the quadratic constraints equation (10.17) results in a quite complicated system of 7 polynomial equations in 7 unknowns $\alpha_1, \ldots, \alpha_6, \beta$. Therefore, we first simplify these equations by eliminating the unknown time shift $\beta$ from these equations using the elimination ideal method presented in [71]. This results in a system of 18 equations in 6 unknowns $\alpha_1, \ldots, \alpha_6$. Even though this system contains more equations than the original system, its structure is less complicated. We solve this system using the automatic generator of Gröbner basis solvers [65]. The final Gröbner basis solver performs Gauss-Jordan elimination of a $194 \times 210$ matrix and the eigenvalue computations of a $16 \times 16$ matrix, since the problem has 16 solutions. Note that by simply applying [65] to the original system of 7 equations in 7 unknowns a huge and numerically unstable solver of size $633 \times 649$ is obtained.

### 10.2.2 Generalized eigenvalue solution to epipolar geometry

Using the non-minimal number of nine point correspondences, the epipolar constraint equation (10.11) can be rewritten as

$$(\mathtt{M}_1 + \beta \mathtt{M}_2)\mathbf{f} = \mathbf{0}, \tag{10.18}$$

where $\mathtt{M}_1$ and $\mathtt{M}_2$ are $9 \times 9$ coefficient matrices and $\mathbf{f}$ is a vector containing nine elements of the fundamental matrix $\mathtt{F}$

The formulation equation (10.18) is a generalized eigenvalue problem (GEP) for which efficient numerical algorithms are readily available. The eigenvalues of equation (10.18) give us solutions to $\beta$ and the eigenvectors to fundamental matrix $\mathtt{F}$.

For this problem the rank of the matrix $\mathtt{M}_2$ is only six and three from nine eigenvalues of equation (10.18) are always zero. Therefore, instead of $9 \times 9$ we can solve only $6 \times 6$ GEP.

This generalized eigenvalue solution is more efficient than the minimal solution presented in section 10.2, however note that the GEP solution uses non-minimal number of nine point correspondences and the resulting fundamental matrix does not necessarily satisfy $\det(\mathtt{F}) = 0$.

### 10.2.3 Minimal solution to homography estimation

The minimal solution to the simultaneous estimation of the homography and the unknown time shift $\beta$ starts with the equations of the form equation (10.13).

First, the solver eliminates the scalar values $\lambda_i$ from equation (10.13). This is done by multiplying equation (10.13) by the skew symmetric matrix $[\mathbf{u}_i + \beta\mathbf{v}_i]_\times$. This leads to the matrix equation

$$[\mathbf{u}_i + \beta\mathbf{v}_i]_\times \mathtt{H}\mathbf{s}_i = \mathbf{0}. \tag{10.19}$$

The matrix equation equation (10.19) contains three polynomial equations from which only two are linearly independent, because the skew symmetric matrix has rank two. This means that we need at least 4.5 (5) samples in two images to estimate the unknown homography $\mathtt{H}$ as well as the time shift $\beta$.

Now let us use the equations corresponding to the first and second row of the matrix equation equation (10.19). In these equations $\beta$ multiplies only the $3^{rd}$ row of the unknown homography matrix. This lead to nine homogeneous equations in 12 monomials $\mathbf{w} = [h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33}, \beta\, h_{31}, \beta\, h_{32}, \beta\, h_{33}]^\top$ for 4.5 samples in two images (i.e. we use only one equation from the three equations equation (10.19) for the 5th sample).

We can stack these nine equations into a matrix form $\mathtt{M}\,\mathbf{w} = \mathbf{0}$, where $\mathtt{M}$ is a $9 \times 12$ coefficient matrix. Assuming that $\mathtt{M}$ has full rank equal to nine, *i.e.*, we have non-degenerate samples, the dimension of $\mathrm{null}(\mathtt{M})$ is 3. This means that the monomial vector $\mathbf{w}$ can in general be rewritten as a linear combination of three null space basis vectors $\mathbf{n}_i$ of the matrix $\mathtt{M}$ as

$$\mathbf{w} = \textstyle\sum_{i=1}^{3} \gamma_i\, \mathbf{n}_i, \tag{10.20}$$

where $\gamma_i$ are new unknowns. Without loss of generality, we can set $\gamma_3 = 1$ to fix the scale of the homography and to bring down the number of unknowns. For 5 or more samples, instead of null space vectors $\mathbf{n}_i$, we use in equation (10.20) three right singular vectors corresponding to three smallest singular values of $\mathtt{M}$.

The elements of the monomial vector $\mathbf{w}$ are not independent. We can see that $w_{10} = \beta\, w_7, w_{11} = \beta\, w_8$, and $w_{12} = \beta\, w_9$, where $w_i$ is the $i^{th}$ element of the vector $\mathbf{w}$. These three constraints, together with the parametrization from equation (10.20) form a system of three quadratic equations in three unknowns $\gamma_1, \gamma_2$, and $\beta$ and only 6 monomials. This system of three equations has a very simple structure and can be directly solved by performing G-J elimination of the $3 \times 6$ coefficient matrix $\mathtt{M}_1$ representing these tree polynomials, and then by computing eigenvalues of the $3 \times 3$ matrix obtained from this eliminated matrix $\mathtt{M}_1$. This problem results in up to three real solutions.

Note, that the problem of estimating homography and $\beta$ can also be formulated as a generalized eigenvalue problem, similarly as the problem of estimating epipolar geometry (Section 10.2.2). However, due to the lack of space and the fact that the presented minimal solution is extremely efficient, we do not describe the GEP homography solution here.

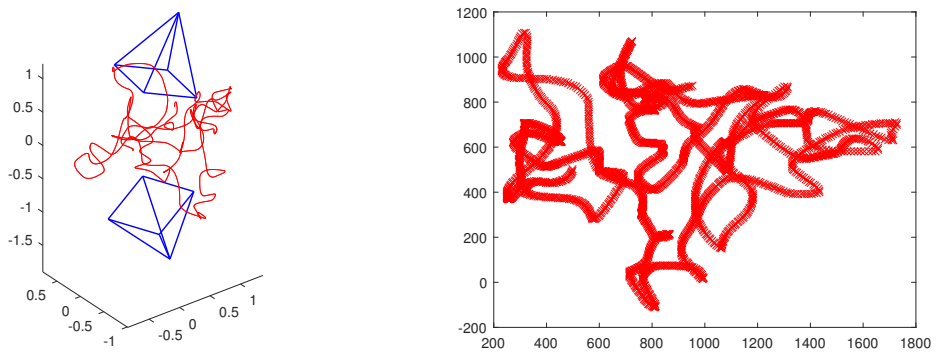


Figure 10.3: An example of the randomly generated scene for the synthetic experiments. On the left is the 3D trajectory with cameras and on the right is an image projected into one of the cameras.

## 10.3 Using RANSAC

In this section we would like to emphasize the role of RANSAC for our solvers. RANSAC is generally used for robustness since the minimal solvers are sensitive to noise and outliers. Outliers in the data will usually come from two sources. One are the mismatches and misdetections and the other is the non-linearity of the point trajectory. Even without gross outliers due to false detections, there will always be outliers with respect to the model in places where the trajectory is not straight on the interpolating interval. Therefore, it is usually beneficial to use RANSAC even if we are sure the correspondences are precise.

By using RANSAC, we avoid those parts of the trajectory and pick the parts that are approximately straight and linear in velocity. Basically we only need to sample 8(F) or 5(H) parts of the trajectory where this assumption holds to obtain a good model, even if the rest of the trajectory is highly non-linear.

## 10.4 Performance of the solvers on synthetic data

First, we investigated the performance of estimating the time shift $\beta$ using the proposed `F` and `H` minimal solvers. We simulated a random movement of a 3D point in front of two cameras. The simulated 3D trajectory was then sampled at different times in each camera, the difference being the ground truth time shift $\beta_{gt}$. Image noise was added from a normal distribution with $\sigma = 0.5$ px. We tested the minimal solvers with various interpolation distances $d$ and compared them also to the standard seven point fundamental matrix (7pt-F) and four point homography (4pt-H) solvers [44]. Each algorithm was tested on 100 randomly generated scenes for each $\beta_{gt}$, resulting in tens of thousands of experiments. Figure 10.3 shows an example of a randomly generated scene for the synthetic experiments.

There are multiple observations we can make from the results. The main one is that both `F` and `H` solvers perform well in terms of estimating $\beta_{gt}$, even for the minimal interpolation distance

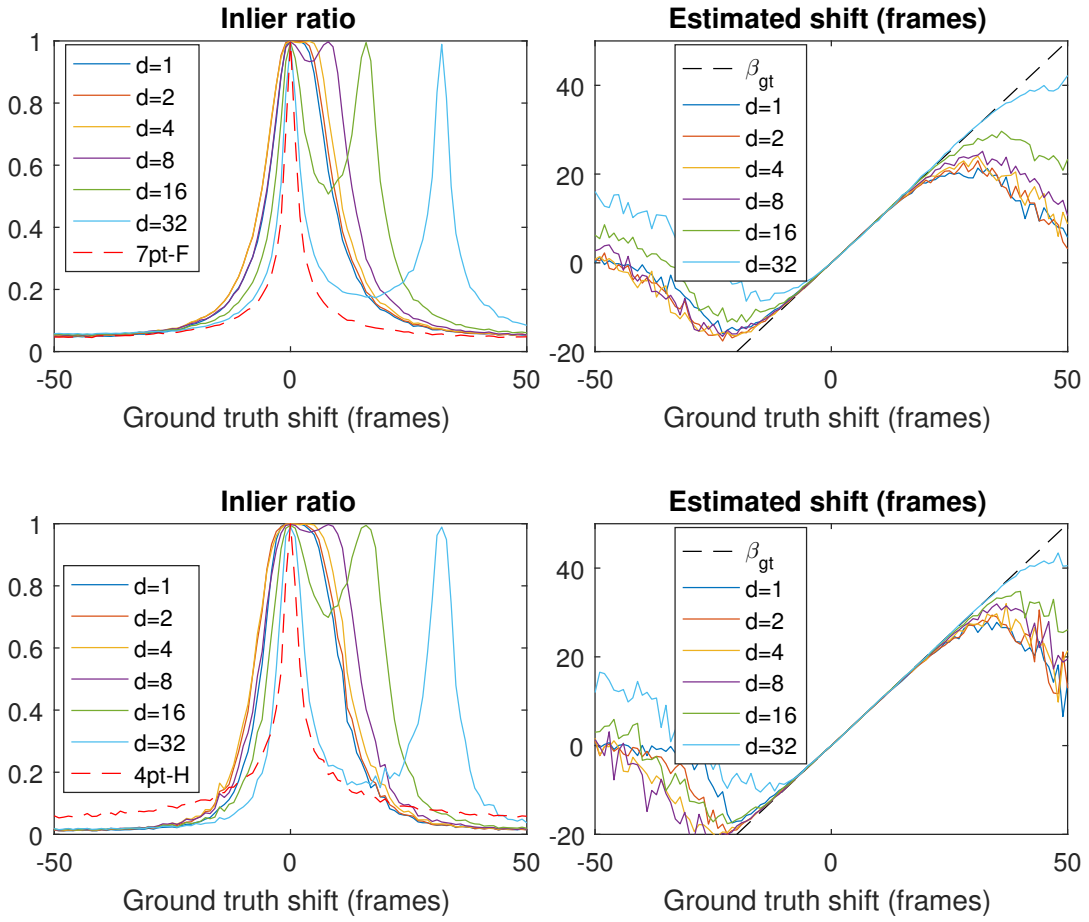


Figure 10.4: Results on randomly generated scene with various time shift $\beta$ between cameras and several different interpolation distances $d$. Temporal distance of one frame equals to approximately 8 pixels distance in 1000x1000px image. Top two figures are results for epipolar matrix and bottom two for homography.

$d = 1$. Figure 10.4 shows that almost all inliers are correctly classified using $d = 1, d = 2, d = 4$ up to shift of 5 frames forward. Furthermore, even though the inlier ratio begins to decrease with larger shifts, time shift $\beta$ is still correctly estimated, up till frame shifts of 20. Overall, for a given $d$, each algorithm was able to estimate correct $\beta$ at least up to $d$. This is a nice property, suggesting that for larger time shifts we should be able to estimate them simply by increasing $d$.

For $d = 8, d = 16, d = 32$, the situation is slightly different with respect to inliers. Notice that there are two peaks in the number of inliers, one at $\beta_{gt} = 0$ and the other at $\beta_{gt} = d$. This is expected, because at $\beta_{gt} = d$, the interpolating vector $\mathbf{v}$ passes through the sample $\mathbf{s}'_{i+\beta_{gt}}$ which is in temporal correspondence with $\mathbf{s}_i$. When $\beta_{gt} \neq 0$ our solvers are for any $d$ well above the number of inliers provided by standard `F` and `H` algorithms.

Another thing to notice is the non-symmetricity of the results. Obviously, when $\beta_{gt} < 0$

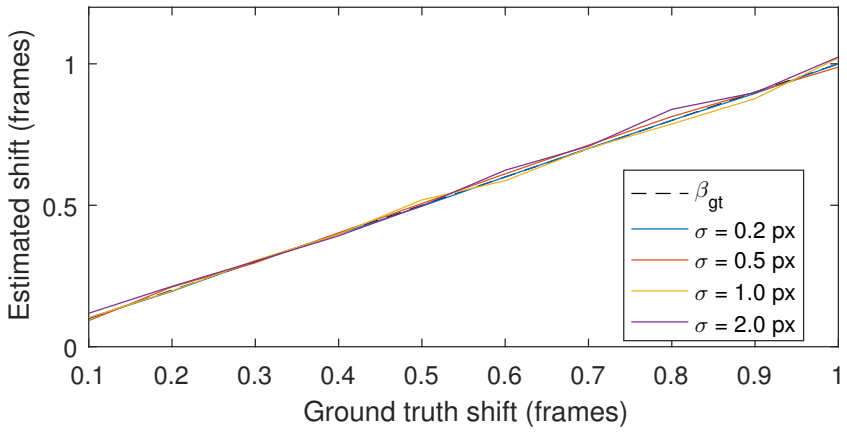


Figure 10.5: Subframe time shift estimation using the fundamental matrix solver. The solver was tested with different levels of image noise.

(backward) and we are interpolating with $d$-th (forward) sample, the peaks in inliers are not present, since we will never hit the sample which is in correspondence. Also, the performance in terms of inliers is reduced when interpolating in the wrong direction, although still above the algorithms not modelling the time shift. Estimation of $\beta$ deteriorates significantly sooner for negative $\beta_{gt}$, at around -10 frames. We will show how to overcome this non-symmetricity by searching over $d$ in both directions using an iterative algorithm.

### 10.4.1 Subframe synchronization

An interesting issue is the ability of the solvers to synchronize sub-frame time shifts, *i.e.*, shifts where $\beta_{gt}$ is not an integer. In the real datasets, images were either hardware synchronized, *i.e.*, $\beta_{gt} = 0$, or we did not have precise enough ground truth information about the subframe time shift. Therefore, we tested the subframe synchronization on the synthetic data only. The results in Figure 10.5 show that the subframe synchronization is very precise for various levels of noise.

### 10.4.2 Accuracy of the estimated geometry

On the same data as used in section 10.4, we evaluated the estimated relative rotations $\mathtt{R}$ and translations $\mathbf{t}$. The results in Figure 10.6 show that we are able to estimate $\mathtt{R}$ and $\mathbf{t}$ significantly better than the classical 7 point algorithm. The utility of our solver is especially apparent from the zoomed in figures with smaller time shifts. The error in $\mathtt{R}$ and $\mathbf{t}$ is almost zero up to 5 frames shift, for shorter interpolation distances $d = 1, 2, 4, 8$. In contrast, such shift causes a significant drop in performance of the classical 7-point algorithm, resulting in errors up to 5 degrees in orientation and relative error of 5% in the translation vector.

Even for the long interpolation distances $d = 16, 32$—although not as good as for $d = 1, 2, 4, 8$—the performance is still better than that of the classical 7-point algorithm. The performance of $d = 16$ and $d = 32$ improves with increasing ground truth time shift and peaks, as

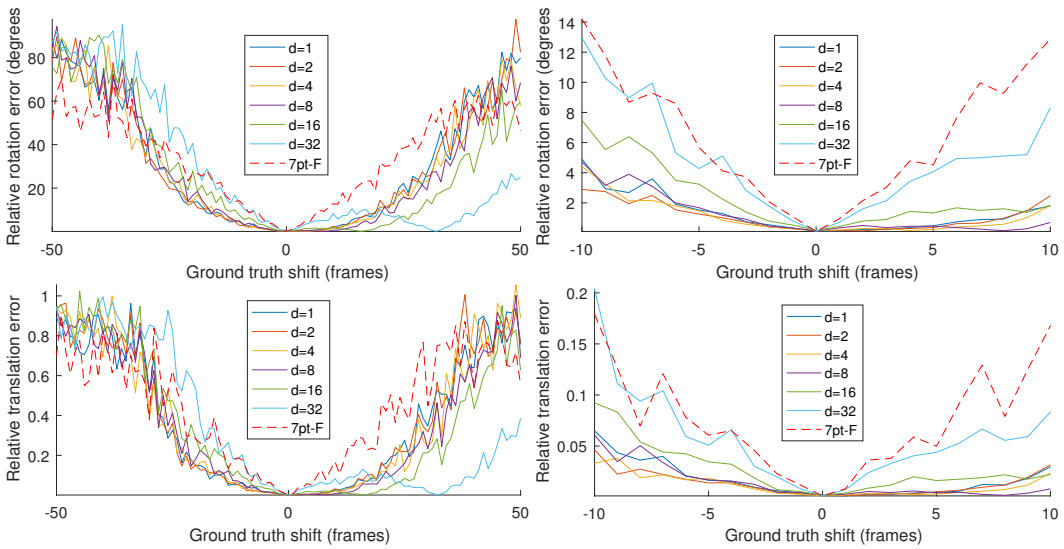


Figure 10.6: Error in the relative rotation and translation between the two cameras from synthetic data extracted from the computed fundamental matrix. Our solvers provide significantly better rotation and translation estimates than the classic 7-point algorithm. Note that in our iterative algorithm, we only use the right hand side of the results in above graphs, because both $d$ and $-d$ are used at each iteration.

expected, on time shifts 16 and 32, respectively. Note that in our iterative algorithm, we only use the right hand side of the results in the above graphs, because both $d$ and $-d$ are used at each iteration.

## 10.5 Iterative algorithm

As we observed in the synthetic experiments, the performance of the minimal solvers will depend on the distance from the optimum, i.e. the distance between the initial estimate $\beta_0$ and the true time shift $\beta_{gt}$, and on the distance $d$ of the samples used for interpolation. The results from synthetic experiments (Figure 10.4) provide useful hints on how to construct an iterative algorithm to improve the performance and applicability of the minimal solvers. In particular, there are three key observations to consider.

First, the number of inliers obtained from RANSAC seems to be a reasonable function to optimize. Generally it will have two strong local maxima, one at $t_i = t'_i$ and one at $(t_i - t'_i) = d$. At $t_i = t'_i$ the sequences are synchronized and at $(t_i - t'_i) = d$, Fig. 10.4, we obtain the correct $\beta$. Both situations give us synchronized sequences. Second, the $\beta$ computed even far from the optimum, although not precise, provides often a good indicator of the direction towards $t_i = t'_i$. Finally, it can be observed that increasing $d$ improves the estimates when we are far from the optimum. Moreover, as seen from the peaks in Fig. 10.4, selecting larger $d$ yields increasingly

better estimates of $\beta$, which are lower or equal than the actual $(t_i - t'_i)$, but never higher. This suggests that we could safely increase $d$ until a better estimate is found.

The observations mentioned above lead us to algorithm 1. The basic principle of the algorithm is the following. In the beginning, assume $i = j$. At each iteration $k$, estimate $\beta$ and `F`. If this model gives more inliers than previous estimate, change $j$ to the nearest integer to $j + \beta$ and repeat. If the new estimate gives less inliers than the last one, extend the search direction by increasing $d$ by powers of 2 until more inliers are found. If $d^{p_{max}}$ is reached, $p$ is reset to 0, so interpolation distances keep circling between $d^0$ and $d^{p_{max}}$. This is essentially a line search over the parameter $d$. Algorithm is stopped when the number of inliers did not increase $p_{max}$ times. This ensures, that at each $t'_j$, all interpolation distances are tested at maximum once. The resulting estimate of $\beta$ is then $j - i + \beta$, which is the difference in frames the algorithm traveled plus the last estimate of time shift at this point (subframe synchronization).

Estimating of $\beta$ and `F` is done using RANSAC and interpolating from both the next and previous $d^{th}$ sample, searching the space of $\beta$ in both directions. Whichever direction returns more inliers is taken as current estimate. By changing the values $p_{min}$ and $p_{max}$ we have the option to adjust the range of search. Having an initial guess about the amount of time shift, e.g. not more than 100 frames, but definitely more than 10 frames, we could start the algorithm with values $p_{min} = 3$ and $p_{max} = 7$ so the search in $d$ would start with $d = 8$ and not go further than $d = 128$.

The symbol `T` represents a geometric relation, in our case either a fundamental matrix or a homography.

## 10.6 Iterative algorithm visualization

In Figure 10.8, we provide a visualization of one run of the iterative algorithm with $p_{max} = 5$. Each iteration is marked by a black square and denoted by the number of the iteration $k$ and the distance $d$ used for interpolation in the given iteration. The algorithm greedily searches for a larger number of inliers (the top figure) and uses the estimated $\beta_k$ to change the correspondences, which results in change of the current ground truth shift (bottom figure). This particular run converged in 6 iterations, even though the initial time shift (50) was larger than the maximum interpolation distance $d = 32$. Moreover, the algorithm only used interpolation distances $d = 1, 2$. These distances were enough to provide a good enough estimate of the time shift that lead to an increased number of inliers.

## 10.7 Real data experiments

Our real datasets contain two private datasets and three publicly available multi-camera datasets. We aimed at collecting various types of scenes to cover wide range of applications. The public data were always synchronized and we manually shifted the frame to frame correspondences to simulate the ground truth time shift. We experimented with shift of -50 to 50 frames on each dataset, which produced time shifts ranging from 2s to 5s based on the camera framerate.

**Algorithm 1** Iterative sync

**Input:** $\mathbf{s}_0, \ldots, \mathbf{s}_n$,$\mathbf{s}'_0, \ldots, \mathbf{s}'_{n'}$,$k_{max}$,$p_{max}$,$p_{min}$
**Output:** $\beta$,$\mathtt{T}$
  $\beta_0 \leftarrow 0$,$i = j$,*skipped* $\leftarrow 0$, $d \leftarrow 2^{p_{min}}$,*inliers*$_0 \leftarrow 0$,$p \leftarrow p_{min}$
  **while** $k = 1 < k_{max}$ **do**
    $\mathtt{T}_1$,$\beta_1$ and *inliers*$_1 \leftarrow$ RANSAC($\mathbf{s}_i, \mathbf{s}'_j, d$)
    $\mathtt{T}_2$,$\beta_2$ and *inliers*$_2 \leftarrow$ RANSAC($\mathbf{s}_i, \mathbf{s}'_j, -d$)
    **if** *inliers*$_1 >$ *inliers*$_2$ **then**
      *inliers*$_k \leftarrow$ *inliers*$_1$, $\beta_k \leftarrow \beta_1$, $\mathtt{T}_k \leftarrow \mathtt{T}_1$
    **else**
      *inliers*$_k \leftarrow$ *inliers*$_2$, $\beta_k \leftarrow \beta_2$, $\mathtt{T}_k \leftarrow \mathtt{T}_2$
    **end if**
    **if** *skipped* $> p_{max}$ **then**
      **return** $\mathtt{T}_{k-1}$,$\beta \leftarrow j - i + \beta_{k-1}$
    **else if** *inliers*$_k <$ *inliers*$_{k-1}$ **then**
      **if** $p < p_{max}$ **then**
        $p \leftarrow p + 1$
      **else**
        $p \leftarrow 0$
      **end if**
      $d \leftarrow 2^p$
      *skipped* $\leftarrow$ *skipped* $+ 1$
    **else**
      $j \leftarrow j + \lceil \beta_k \rfloor$
      *skipped* $\leftarrow 0$
      $k \leftarrow k + 1$
    **end if**
  **end while**

### 10.7.1 Datasets

Dataset Marker was obtained by moving an Aruco marker in front of a two webcams running at 10fps. A digital clock in the scene was processed by OCR in each frame to provide ground truth timestamps. Further, we used three public datasets and one private: UvA [52], KITTI [38], Hockey and PETS [31]. The UvA dataset consists of video sequences taken by three static cameras with manual annotations of humans. The KITTI dataset contains stereo video sequences taken from a moving car. In our experiments, we used raw unsynchronized data provided by the authors. The Hockey dataset was synchronized by [112] and the trajectories are manually curated tracks of [51]. The PETS dataset is a standard multi-target tracking dataset. Trajectories were detected by [39, 33, 99] and manually joined.

### 10.7.2 Algorithms

We compared seven different approaches to simultaneously solving two-camera geometry and time shift. Depending on the data, either fundamental matrix or homography was estimated. We denote both geometric relations by $T$, where $T$ means `H` or `F` was estimated using standard 4 or 7 point algorithms [44] and $T_\beta$ means that `H` or `F` was estimated together with $\beta$. The

| $\beta_{gt}$ | 0-10 | 10-20 | 20-30 | 30-40 | 40-50 |
|---|---|---|---|---|---|
| $T_\beta$-new-iter-pmax0 | 4.7 | 4.3 | 3.5 | 4.1 | 3.8 |
| $T_\beta$-new-iter-pmax6 | 23 | 22 | 21.2 | 21.6 | 21.2 |
| $T_\beta$-new-iter-pmaxvar | 18 | 19 | 17.5 | 16.7 | 16.5 |

Table 10.1: Average number of RANSACs executed before termination. Evaluated on Marker dataset.

rightmost column of figure 10.7 shows which model, i.e. homography or fundamental matrix, was estimated on a particular data set.

The closest alternatives to our approach are the least-squares based algorithms presented in [86] and [84]. Both optimize $\mathtt{F}$ or $\mathtt{H}$ and $\beta$ starting from an initial estimate of $\beta = 0$ and $T$. Method [86] uses linear interpolation from the next sample, whereas method [84] uses spline interpolation of the image trajectory and we will refer to these methods as $T_\beta$-lin and $T_\beta$-spl respectively. In our implementation of those methods, we used Matlab's lsqnonlin function with Levenberg-Marquardt algorithm, all stopping criteria set to epsilon and maximum number of 100 iterations.

We tested the solvers presented in section 10.2 with $d = 1$ as algorithm $T_\beta$-new-d1. The proposed iterative algorithm 1 that uses the solvers was tested using several different settings. The user can control the algorithm using parameters $p_{max}$ and $p_{min}$, which determine the distances $d$ that will be used for interpolation. As we observed in section 10.4, there is a good chance of computing a correct $\beta$ if $d > \beta_{gt}$. First, we ran the algorithm with $p_{min} = 0$ and $p_{max} = 6$, which gives maximum $d = 64$ as algorithm $T_\beta$-new-iter-pmax6. This version of the algorithm is guaranteed to try $d = 1, 2, 4, 8, 16, 32, 64$ at each $\beta_k$ before it stops or it finds more inliers. This covers the time shifts we tested, but can lead to unnecessary iterations for smaller shifts. Therefore, we also tested $p_{max} = 0$ as $T_\beta$-new-iter-pmax0 which only tried $d = 1$ at each iteration to see the capabilities of the most efficient version of the algorithm.

The last version of our algorithm, $T_\beta$-new-iter-pmaxvar, adapted both $p_{max}$ and $p_{min}$ to $\beta_{gt}$ such that $2^{p_{min}} \leq \beta_{gt} < 2^{p_{max}}$. This represents a case when user has a rough estimate about the expected time shift and sets the algorithm accordingly. We remind that setting $p_{min}$ only affects the initial interpolating distance, after reaching $d = 2^{p_{max}}$ the algorithm starts again with $d = 2^0$.

Finally, algorithm $T$-lin [86] also takes the next samples for interpolation, making it comparable to our $T_\beta$-new-d1. We used $T$-lin in the same iterative scheme as $T_\beta$-new-iter-pmax6 and tested it as $T$-new-lin-iter, where instead of using the number of inliers as a criteria for accepting a step, we used the value of the residual.

### 10.7.3 Accuracy of the estimated geometry

The only real world dataset experiments for which the ground truth spatial calibration is provided is the UvA dataset. We extracted the ground truth relative $\mathtt{R}_{gt}$ and $\mathbf{t}_{gt}$ from the dataset camera matrices and compared them to the values estimated by all algorithms. Figure 10.9 shows the angular error of $\mathtt{R}$, measured as the rotation angle of $\mathtt{R}_{err} = \mathtt{R}^\top \mathtt{R}_{gt}$, and the relative translation

error measured as $||\mathbf{t}_{gt} - \mathbf{t}||$, where both $\mathbf{t}_{gt}$ and $\mathbf{t}$ are normalized to unit lengths. Errors are averaged over 100 runs for each datapoint.

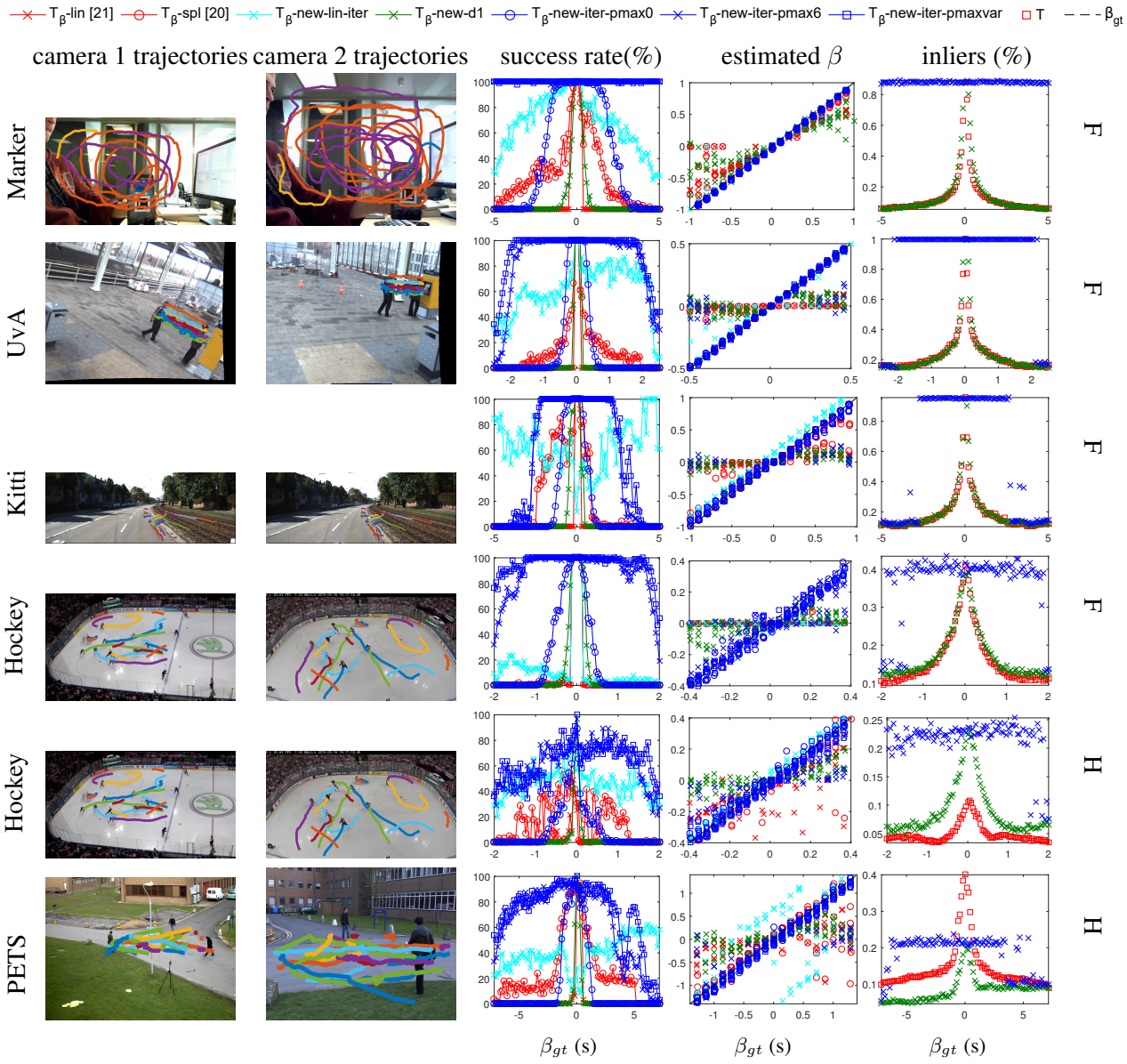


Figure 10.7: Results on real data. In the two leftmost columns, trajectories used for the computations are depicted in coloured lines over a sample images from the dataset. Third column shows the rates with which different algorithms succeeded to synchronize the sequence to single frame precision for various ground truth time shifts. Fourth column shows a closer look at the individual results for $\beta$ for smaller ground truth time shifts and five runs for each algorithm, each data point corresponds to one run of the algorithm at corresponding $\beta_{gt}$. Letters H and F on the right signalize whether homography or fundamental matrix was computed.

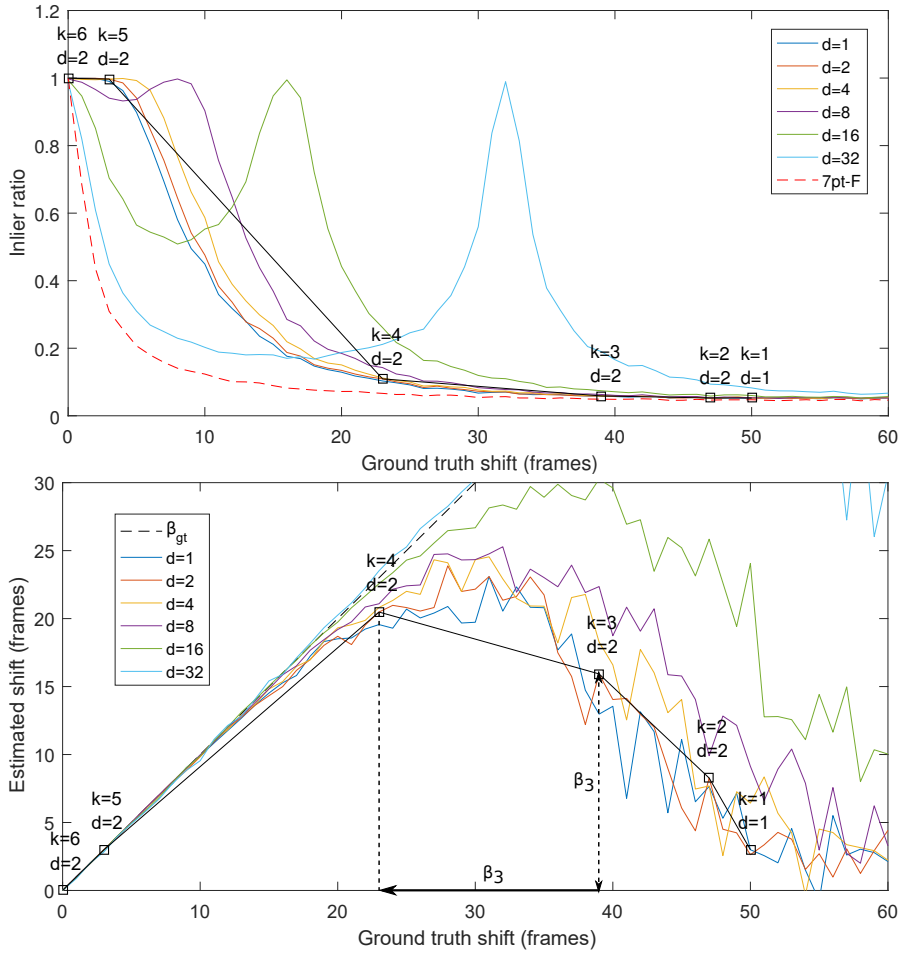


Figure 10.8: An example of one run of the iterative algorithm. $k$ is the iteration number and $d$ is the interpolation distance used. Beginning with time shift of 50 frames, the algorithm would converge in 6 iterations.

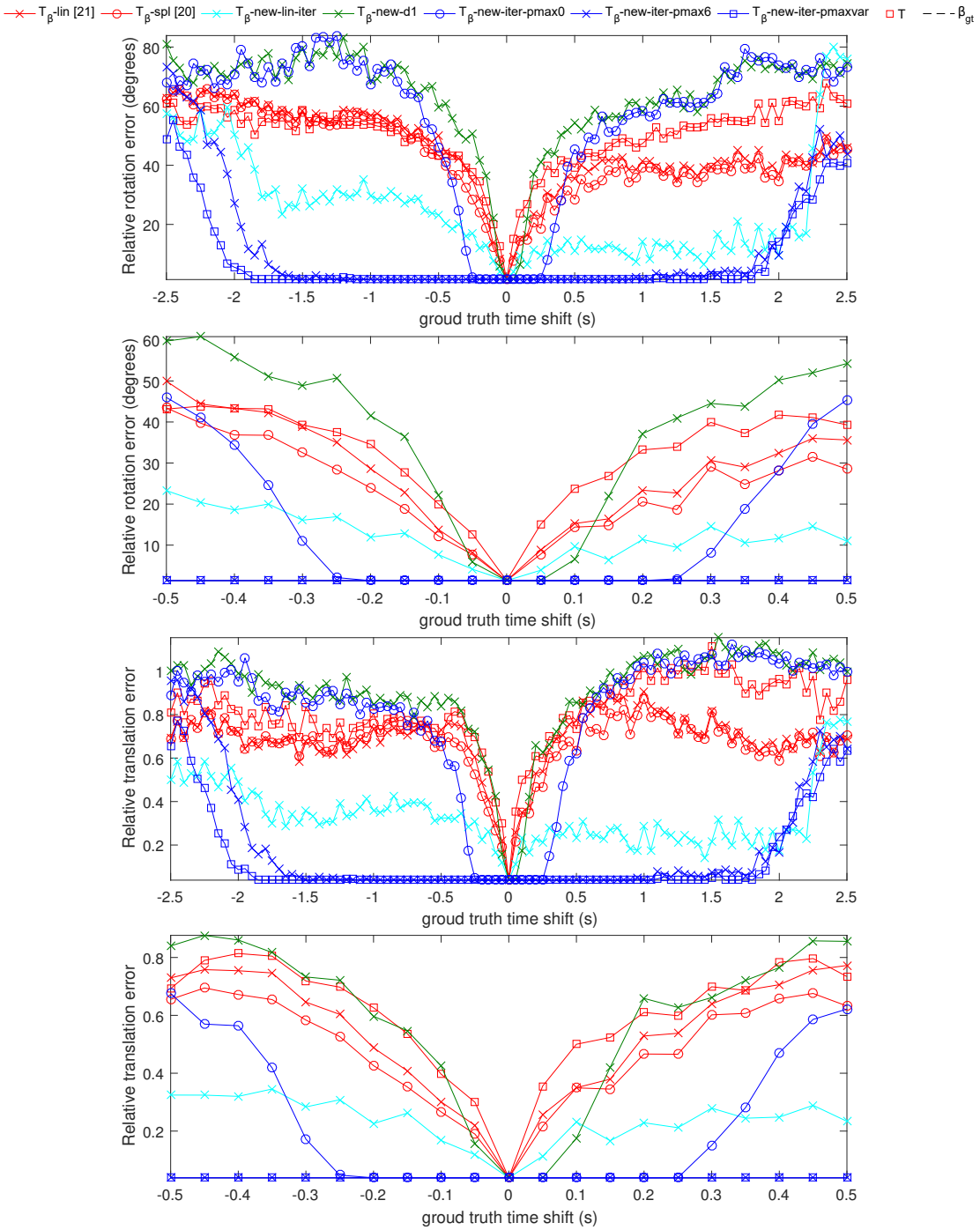


Figure 10.9: Error in relative rotation and translation between the two cameras from UvA dataset. All algorithms were tested, taking the resulting fundamental matrix and decomposing it into $\mathtt{R}$ and $\mathbf{t}$.

The results follow the pattern of the results in Figure 4, where when an algorithm successfully estimated the time shift, it also provided a good geometry estimate.

Both iterative algorithms $T_\beta$-new-iter-pmax6 and $T_\beta$-new-iter-pmaxvar, which have $pmax$ large enough to cover the required time shifts, perform well over almost the entire range of time shifts. The efficient $T_\beta$-new-iter-pmax0 which iteratively uses $d = 1$ performed well up till the time shifts of 0.25 s (5 frames). $T_\beta$-new-d1, which is the solver using $d = 1$ in RANSAC, was able to estimate the geometry reliably only for a time shift of 1 frame. All the algorithms based on the 7-point algorithm, including the 7-point algorithm itself in RANSAC, performed poorly on this dataset.

### 10.7.4 Results and discussion

The results on real datasets demonstrate a wide practical usefulness of the proposed methods. For most datasets, $T_\beta$-new-d1 itself performed at least as good as the least squares algorithms $T_\beta$-lin and $T_\beta$-spl. A single RANSAC was enough to synchronize time shifts of 2-5 frames across all datasets. The iterative algorithm $T_\beta$-new-iter-pmax6 built upon our solvers performed the absolute best across all datasets, converging successfully from as far as 5s time difference on Marker and Hockey datasets, 2s difference on UvA dataset and 2.5s on Kitti dataset as seen in the success rate column of figure 10.7.

On the Kitti dataset, $T_\beta$-new-iter-pmax6 was outperformed by the $T_\beta$-new-lin-iter, which uses the iterative algorithm proposed by us, but with a solution from [86] inside. $T_\beta$-new-lin-iter was able to estimate time differences larger than 2.5s but only in roughly half of the cases, where $T_\beta$-new-iter-pmax6 was 100% successful up to 2.5s when it sharply fell off. We account this to the high non-linearity of the 2D velocity of the image points, where as the objects got closer to the car, they moved faster. The tracks of length 25 frames and more were very sparse here and the longer they were the more non-linear in the velocity.

On the contrary, the hockey dataset posed a big challenge for the least squares algorithms, which struggled even with the smallest time offsets. We attribute this to the poor estimate of `F` by the seven point algorithm which causes the LM algorithm to get stuck in local minima. We also tested the homography version of all algorithms on this dataset, since the trajectories are approximately planar, which resulted in the least squares algorithms performing slightly better whereas the algorithms with minimal solvers performed slightly worse.

PETS dataset was probably the most challenging, because of the low framerate (7FPS), coarse detections and abrupt change of motion. Still, our methods managed to synchronize the sequences in majority of cases.

Table 10.1 shows the average number of RANSACs used before termination of different variants of iterative algorithm 1 for the dataset Marker. We can see that using 8pt-iter-pmax0 greatly reduces the computations needed, still allowing this method to reliably estimate time shifts of 0.5s-2s depending on the scene, rendering it useful if we are certain that the sequences are off by only a few tens of frames. Knowing the time shift approximately and setting $p_{max}$ and $p_{min}$ can also reduce the computations as shown by 8pt-iter-pmaxvar, which provided identical performance to 8pt-iter-pmax6, sometimes even outperforming it.

## 10.8 Conclusion

We have presented solvers for simultaneously estimating epipolar geometry or homography and time shift between image sequences from unsynchronized cameras. These are the first minimal solutions to these problems, making them suitable for robust estimation using RANSAC. Our methods need only trajectories of moving points in images, which are easily provided by state-of-the-art methods, e.g. SIFT matching, human pose detectors, or pedestrian trackers. We were able to synchronize wide range of real world datasets shifted by several frames using a single RANSAC with our solvers. For larger time shifts, we proposed an iterative algorithm using these solvers in succession. The iterative algorithm proved to be reliable enough for synchronizing real world camera setups ranging from autonomous cars, surveillance videos, and sport game recordings, which were de-synchronized by several seconds.

# 11 Conclusions

This thesis solved several essential problems in 3D computer vision that arise when the geometry of the scene changes due to camera or object motion. Adapted algorithms that describe the geometry of moving rolling shutter cameras and unsynchronized stereo setups that observe motion were presented. Issues arising when employing the new camera models were discussed and possible solutions proposed.

In particular, several algorithms for rolling shutter camera absolute pose were presented. A first minimal solution to rolling shutter absolute pose from six point correspondences, called R6P, is based on the double-linearized model, is quite efficient (300$\mu$s), but it requires initialization of the camera orientation, e.g., from P3P.

A stand-alone solution to R6P, using a single-linearized model that does not need an initialization to provide the camera pose and motion is presented as well. It uses Cayley parameterization of the camera orientation and leads to a much more difficult system of equations. Its runtime is, therefore, slower, around 1.4ms, but still viable for robust estimation using RANSAC or real-time applications such as augmented reality.

A version of rolling shutter absolute pose problem, that only requires five point correspondences, but also a measurement of the upward (or downward) facing vector is presented next. This version is much more practical for devices that can provide the gravity measurement (up-vector) such as almost every smartphone, new digital cameras or UAV's since it does not require any initialization and runs in only $140\mu s$.

In all these solutions to rolling shutter absolute pose, a simplified system of polynomial equations was formed and solved using the automatic solver generators [65, 74] based on Gröbner bases theory.

Further improvement in practical usability has been achieved by making the equations completely linear by separating the absolute pose estimation into two sets of parameters, alternating between both. This way only simple computations of linear systems are required. In particular, the best performing iterative solver achieves almost identical accuracy to R6P in only two iterations, reducing the computation time to $10\mu$s.

A general rolling shutter bundle adjustment method for unordered sets of images is investigated. It was pointed out that using double-linearized model, any pair of images can be explained by RS cameras and planar scene, moreover, even when using the non-linearized motion models, we will get very close to this particular case. This is a degenerate configuration introduced by the additional parameters describing camera rotation. Not only the planar degeneracy exists, but also is a local minimum with lower error than the true scene configuration; therefore RS BA will always prefer this solution. For three or more cameras, it was shown that only configurations with identical or similar readout directions will collapse into a plane during BA. We have

demonstrated on synthetic and real data that capturing images with readout angles being further apart than 30 degrees will prevent this degeneracy in practice.

New algorithms for two-camera geometry with un-synchronized image sequences were introduced. A minimal, 8-point algorithm to compute the camera fundamental matrix and time shift between sequences was introduced. This solution was possible thanks to applying the elimination ideal method presented in [71] and the automatic solver generator [65]. A more efficient, but non-minimal solution from 9 points formulated as a generalized eigenvalue problem is also shown. Further, a 5-point algorithm for homography and time shift is proposed, which can be derived directly without [65] by solving a system of three quadratic equations.

An iterative algorithm, using the minimal solvers and RANSAC to find large time offsets between image sequences was proposed and verified on publicly available real datasets. The method does not require any particular image content, just arbitrary detections of moving objects in the scene. We have shown that we can synchronize pairs of cameras from movement in the image sequences even for large offsets such as several seconds and at the same time we can achieve sub-frame synchronization.

# 11 Bibliography


[1] S. Agarwal, K. Mierle, and Others. Ceres Solver. 2010.

[2] O. Ait-aider, N. Andreff, J. M. Lavest, University 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.

[3] O. Ait-Aider and F. Berry. Structure and kinematics triangulation with a rolling shutter stereo rig. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1835–1840, Sept. 2009.

[4] C. Albl, Z. Kukelova, A. W. Fitzgibbon, J. Heller, M. Smid, and T. Pajdla. On the Two-View Geometry of Unsynchronized Cameras. In *CVPR*, pages 5593–5602, 2017.

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

[6] C. Albl, Z. Kukelova, and T. Pajdla. Rolling shutter absolute pose problem with known vertical direction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3355–3363, 2016.

[7] C. Albl, Z. Kukelova, and T. Pajdla. Rolling shutter camera absolute pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, submitted 2018, first revision - minor.

[8] C. Albl, Z. Kukelova, A. Sugimoto, and T. Pajdla. Linear solution to the minimal absolute pose rolling shutter problem. In *Asian Conference on Computer Vision (ACCV 2018)*, page Submitted 2018, 2018.

[9] C. Albl, A. Sugimoto, and T. Pajdla. Degeneracies in rolling shutter sfm. In *European Conference on Computer Vision*, pages 36–51. Springer, 2016.

[10] M.-a. Ameller, B. Triggs, and L. Quan. Camera pose revisited: New linear algorithms. *In 14eme Congres Francophone de Reconnaissance des Formes et Intelligence Artificielle. Paper in French*, page 2002, 2002.

[11] K. B. Atkinson. *Close Range Photogrammetry and Machine Vision*. Whittles Publishing, Roseleigh House, Latheronwheel, Caithness, Scotland, 1996.

[12] J. C. Bazin and M. Pollefeys. 3-line RANSAC for orthogonal vanishing point detection. In *IROS*, pages 4282–4287, 2012.

[13] R. Boada Farràs. Active camera calibration for robotic systems. 2016.

[14] D. Brown. Close-Range Camera Calibration. *Photogrammetric Engineering*, 37(8):855–&, 1971. WOS:A1971K177500004.

[15] D. C. Brown. The bundle adjustment - progress and prospects. In *XIIIth Congress of the International Society for Photogrammetry*, 1976.

[16] M. Bujnak, Z. Kukelova, and T. Pajdla. A general solution to the p4p problem for camera with unknown focal length. In *Computer Vision and Pattern Recognition (CVPR)*, 2008.

[17] M. Bujnak, Z. Kukelova, and T. Pajdla. New efficient solution to the absolute pose problem for camera with unknown focal length and radial distortion. In *Asian Conference on Computer Vision (ACCV)*, 2010.

[18] M. Bujnak, Z. Kukelova, and T. Pajdla. Making minimal solvers fast. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1506–1513, June 2012.

[19] M. Byrod, K. Josephson, and K. Astrom. Improving numerical accuracy of gröbner basis polynomial equation solvers. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference On*, pages 1–8. IEEE, 2007.

[20] G. Carrera, A. Angeli, and A. J. Davison. Lightweight SLAM and Navigation with a Multi-Camera Rig. In *ECMR*, pages 77–82, 2011.

[21] Y. Caspi, M. Irani, and M. I. Yaron Caspi. Spatio-temporal alignment of sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(11):1409–1424, 2002.

[22] Cheng Lei and Yee-Hong Yang. Tri-focal tensor-based multiple video synchronization with subframe optimization. *IEEE Transactions on Image Processing*, 15(9):2473–2480, Sept. 2006.

[23] O. Chum, J. Matas, and J. Kittler. Locally Optimized RANSAC. In B. Michaelis and G. Krell, editors, *Pattern Recognition: 25th DAGM Symposium, Magdeburg, Germany, September 10-12, 2003. Proceedings*, pages 236–243. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[24] M. A. R. Cooper and P. A. Cross. STATISTICAL CONCEPTS AND THEIR APPLICATION IN PHOTOGRAMMETRY AND SURVEYING. *The Photogrammetric Record*, 12(71):637–663, 1988.

[25] D. A. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*. Graduate Texts in Mathematics. Springer-Verlag, New York, 2 edition, 2005.

[26] C. Dai, Y. Zheng, and X. Li. Subframe video synchronization via 3D phase correlation. In *International Conference on Image Processing*, pages 501–504, 2006.

[27] Y. Dai, H. Li, and L. Kneip. Rolling Shutter Camera Relative Pose: Generalized Epipolar Geometry. *arXiv:1605.00475 [cs]*, May 2016.

[28] A. M. Danilevskii. The numerical solution of the secular equation. *Matematicky Sbornik*, (44(2)):169–171, 1937.

[29] J. Deutscher and I. Reid. Articulated body motion capture by stochastic search. *International Journal of Computer Vision*, 61(2):185–205, 2005.

[30] A. Elhayek, C. Stoll, N. Hasler, K. I. Kim, H.-P. Seidel, and C. Theobalt. Spatio-temporal Motion Tracking with Unsynchronized Cameras. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1870–1877, Providence, RI, June 2012. IEEE.

[31] A. Ellis, A. Shahrokni, and J. Ferryman. PETS 2009 Benchmark Data. 2009.

[32] O. D. Faugeras, Q. T. Luong, and S. J. Maybank. Camera self-calibration: Theory and experiments. In G. Sandini, editor, *Computer Vision — ECCV'92*, Lecture Notes in Computer Science, pages 321–334. Springer Berlin Heidelberg, 1992.

[33] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, Sept. 2010.

[34] 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.

[35] J.-M. Frahm, K. Köser, and R. Koch. Pose estimation for multi-camera systems. In *Joint Pattern Recognition Symposium*, pages 286–293. Springer, 2004.

[36] F. Fraundorfer, P. Tanskanen, and M. Pollefeys. A Minimal Case Solution to the Calibrated Relative Pose Problem for the Case of Two Known Orientation Angles. In *Computer Vision – ECCV 2010*, Lecture Notes in Computer Science, pages 269–282. Springer, Berlin, Heidelberg, Sept. 2010.

[37] P. Furgale, J. Rehder, and R. Siegwart. Unified temporal and spatial calibration for multi-sensor systems. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1280–1286, Nov. 2013.

[38] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[39] R. B. Girshick, P. F. Felzenszwalb, and D. McAllester. Discriminatively Trained Deformable Part Models, Release 5.

[40] S. I. Granshaw. BUNDLE ADJUSTMENT METHODS IN ENGINEERING PHOTOGRAMMETRY. *The Photogrammetric Record*, 10(56):181–207, 1980.

[41] J. A. Grunert. Das pothenotische problem in erweiterter gestalt nebst bber seine anwendungen in der geodasie. *Grunerts Archiv fur Mathematik und Physik*, pages 238–248, 1841.

[42] R. M. Haralick, D. Lee, K. Ottenburg, and M. Nolle. Analysis and solutions of the three point perspective pose estimation problem. In *1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Proceedings*, pages 592–598, June 1991.

[43] R. Hartley and F. Kahl. Critical Configurations for Projective Reconstruction from Multiple Views. *International Journal of Computer Vision*, 71(1):5–47, June 2006.

[44] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[45] N. Hasler, B. Rosenhahn, T. Thormahlen, M. Wand, J. Gall, and H.-P. Seidel. Markerless motion capture with unsynchronized moving cameras. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference On*, pages 224–231. IEEE, 2009.

[46] M. Hazewinkel, editor. *Encyclopaedia of Mathematics*. Springer Netherlands, Dordrecht, 1989.

[47] J. Hedborg, P.-E. Forssén, M. Felsberg, and Erik Ringaby. Rolling shutter bundle adjustment. In *CVPR*, pages 1434–1441, 2012.

[48] 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.

[49] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1106–1112, June 1997.

[50] D. Henrion, J.-B. Lasserre, and J. Löfberg. GloptiPoly 3: Moments, optimization and semidefinite programming. *Optimization Methods and Software*, 24(4-5):761–779, Oct. 2009.

[51] J. a. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7575 LNCS(PART 4):702–715, 2012.

[52] M. Hofmann and D. M. Gavrila. Multi-view 3D Human Pose Estimation Combining Single-frame Recovery, Temporal Integration and Model Adaptation. In *CVPR*, pages 2214–2221, 2009.

[53] D. G. Hook and P. R. McAree. *Using Sturm Sequences to Bracket Real Roots of Polynomial Equations*. Academic Press, Jan. 1990.

[54] E. Ito and T. Okatani. Self-Calibration-Based Approach to Critical Motion Sequences of Rolling-Shutter Structure From Motion. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[55] Y. Jeong, D. Nister, D. Steedly, R. Szeliski, and I.-S. Kweon. Pushing the envelope of modern methods for bundle adjustment. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 1474–1481, 2010.

[56] 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.

[57] K. Josephson and M. Byröd. Pose estimation with radial distortion and unknown focal length. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[58] F. Kahl and R. Hartley. Critical Curves and Surfaces for Euclidean Reconstruction. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *Computer Vision — ECCV 2002*, number 2351 in Lecture Notes in Computer Science, pages 447–462. Springer Berlin Heidelberg, May 2002.

[59] M. Kalantari, A. Hashemi, F. Jung, and J.-P. Guédon. A New Solution to the Relative Orientation Problem Using Only 3 Points and the Vertical Direction. *Journal of Mathematical Imaging and Vision*, 39(3):259–268, 2010.

[60] G. Klein and D. Murray. Parallel Tracking and Mapping on a camera phone. In *8th IEEE International Symposium on Mixed and Augmented Reality, 2009. ISMAR 2009*, pages 83–86, Oct. 2009.

[61] L. Kneip, H. Li, and Y. Seo. UPnP: An Optimal O(n) Solution to the Absolute Pose Problem with Universal Applicability. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, number 8689 in Lecture Notes in Computer Science, pages 127–142. Springer International Publishing, Sept. 2014.

[62] Y. Kuang and K. Åström. Numerically stable optimization of polynomial solvers for minimal problems. In *European Conference on Computer Vision (ECCV)*, 2012.

[63] Z. Kukelova. *Algebraic Methods in Computer Vision*. PhD thesis, Czech Technical University in Prague, 2013.

[64] Z. Kukelova, M. Bujnak, J. Heller, and T. Pajdla. Singly-Bordered Block-Diagonal Form for Minimal Problem Solvers. In D. Cremers, I. Reid, H. Saito, and M.-H. Yang, editors, *Computer Vision – ACCV 2014*, Lecture Notes in Computer Science, pages 488–502. Springer International Publishing, 2015.

[65] Z. Kukelova, M. Bujnak, and T. Pajdla. Automatic Generator of Minimal Problem Solvers. In *ECCV, Part III*, volume 5304 of *Lecture Notes in Computer Science*, 2008.

[66] Z. Kukelova, M. Bujnak, and T. Pajdla. Closed-Form Solutions to Minimal Absolute Pose Problems with Known Vertical Direction. In R. Kimmel, R. Klette, and A. Sugimoto, editors, *Computer Vision – ACCV 2010*, number 6493 in Lecture Notes in Computer Science, pages 216–229. Springer Berlin Heidelberg, Nov. 2010.

[67] Z. Kukelova, M. Bujnak, and T. Pajdla. Polynomial Eigenvalue Solutions to Minimal Problems in Computer Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1381–1393, July 2012.

[68] Z. Kukelova, J. Heller, M. Bujnak, A. Fitzgibbon, and T. Pajdla. Efficient solution to the epipolar geometry for radially distorted cameras. In *International Conference on Computer Vision (ICCV)*, 2015.

[69] Z. Kukelova, J. Heller, B. M., and T. Pajdla. Radial Distortion Homography. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[70] Z. Kukelova, J. Heller, and T. Pajdla. Hand-Eye Calibration without Hand Orientation Measurement Using Minimal Solution. In *11th Asian Conference on Computer Vision (ACCV 2012)*, pages 576–589, Nov. 2013.

[71] Z. Kukelova, J. Kileel, B. Sturmfels, and T. Pajdla. A Clever Elimination Strategy for Efficient Minimal Solvers. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[72] Z. Kukelova, P. Krsek, V. Smutny, and T. Pajdla. Groebner Basis Solutions to Satellite Trajectory Control by Pole Placement. page E81, Sept. 2013.

[73] V. Larsson and K. Åström. Uncovering symmetries in polynomial systems. In *European Conference on Computer Vision (ECCV)*, 2016.

[74] V. Larsson, K. Åström, and M. Oskarsson. Efficient Solvers for Minimal Problems by Syzygy-Based Reduction. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2383–2392, July 2017.

[75] V. Larsson, K. Åström, and M. Oskarsson. Polynomial Solvers for Saturated Ideals. In *International Conference on Computer Vision (ICCV)*, 2017.

[76] G. H. Lee, M. Pollefeys, and F. Fraundorfer. Relative Pose Estimation for a Multi-camera System with Known Vertical Direction. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 540–547, June 2014.

[77] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An Accurate O(n) Solution to the PnP Problem. *International Journal of Computer Vision*, 81(2):155–166, July 2008.

[78] M. Li, B. H. Kim, and A. I. Mourikis. Real-time motion tracking on a cellphone using inertial sensing and a rolling-shutter camera. In *2013 IEEE International Conference on Robotics and Automation*, pages 4712–4719, May 2013.

[79] M. I. A. Lourakis and A. A. Argyros. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Trans. Math. Softw.*, 36(1):2:1–2:30, Mar. 2009.

[80] 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 A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *Computer Vision – ECCV 2012*, number 7572 in Lecture Notes in Computer Science, pages 456–469. Springer Berlin Heidelberg, Oct. 2012.

[81] M. Meingast, C. Geyer, and S. Sastry. Geometric Models of Rolling-Shutter Cameras. *Computing Research Repository*, 2005.

[82] P. Moulon, P. Monasse, and R. Marlet. Global Fusion of Relative Motions for Robust, Accurate and Scalable Structure from Motion. In *2013 IEEE International Conference on Computer Vision (ICCV)*, pages 3248–3255, Dec. 2013.

[83] O. Naroditsky and K. Daniilidis. Optimizing polynomial solvers for minimal geometry problems. In *International Conference on Computer Vision (ICCV)*, 2011.

[84] M. Nischt and R. Swaminathan. Self-calibration of asynchronized camera networks. In *2009 IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 2164–2171, Sept. 2009.

[85] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer-Verlag, New York, 2 edition, 2006.

[86] M. Noguchi and T. Kato. Geometric and Timing Calibration for Unsynchronized Cameras Using Trajectories of a Moving Marker. In *IEEE Workshop on Applications of Computer Vision, 2007. WACV '07*, pages 20–20, Feb. 2007.

[87] F. L. C. Padua, R. L. Carceroni, G. Santos, and K. N. Kutulakos. Linear Sequence-to-Sequence Alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2):304–320, 2010.

[88] D. Pundik and Y. Moses. Video Synchronization Using Temporal Signals from Epipolar Lines. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Computer Vision – ECCV 2010*, Lecture Notes in Computer Science, pages 15–28. Springer Berlin Heidelberg, Sept. 2010.

[89] L. Quan and Z. Lan. Linear N-point camera pose determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):774–780, Aug. 1999.

[90] C. Rao, A. Gritai, M. Shah, and T. Syeda-Mahmood. View-invariant alignment and matching of video sequences. In *IEEE International Conference on Computer Vision*, pages 939–945 vol.2, 2003.

[91] 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.

[92] E. Ringaby and P.-E. Forssén. Efficient Video Rectification and Stabilisation for Cell-Phones. *International Journal of Computer Vision*, 96(3):335–352, 2012.

[93] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer. Speeded up detection of squared fiducial markers. *Image and Vision Computing*, 76:38–47, Aug. 2018.

[94] O. Saurer, K. Koser, J.-Y. Bouguet, and M. Pollefeys. Rolling Shutter Stereo. In *2013 IEEE International Conference on Computer Vision (ICCV)*, pages 465–472, Dec. 2013.

[95] O. Saurer, M. Pollefeys, and G. H. Lee. A minimal solution to the rolling shutter pose estimation problem. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1328–1334, Sept. 2015.

[96] J. L. Schönberger and J.-M. Frahm. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[97] G. Schroth, F. Schweiger, M. Eichhorn, E. Steinbach, M. Fahrmair, and W. Kellerer. Video synchronization using bit rate profiles. In *International Conference on Image Processing*, pages 1549–1552, 2010.

[98] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 519–528. IEEE, 2006.

[99] X. Shi, Z. Yang, and J. Chen. Modified Joint Probabilistic Data Association. In *IEEE International Conference on Computer Vision*, pages 6615–6620, 2015.

[100] K. Shoemake. Animating Rotation with Quaternion Curves. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '85, pages 245–254, New York, NY, USA, 1985. ACM.

[101] P. Shrestha, H. Weda, M. Barbieri, and D. Sekulovski. Synchronization of multiple video recordings based on still camera flashes. In *International Conference on Multimedia*, volume 2, page 137, 2006.

[102] S. Singh, S. A. Velastin, and H. Ragheb. MuHAVi: A Multicamera Human Action Video Dataset for the Evaluation of Action Recognition Methods. In *2010 Seventh IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 48–55, Aug. 2010.

[103] N. Snavely, S. M. Seitz, and R. Szeliski. Photo Tourism: Exploring Photo Collections in 3D. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 835–846, New York, NY, USA, 2006. ACM.

[104] G. Stein. Tracking from multiple view points: Self-calibration of space and time. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 521–527. IEEE Computer Society, 1999.

[105] H. Stewenius. *Gröbner Basis Methods for Minimal Problems in Computer Vision*, volume 2005:1. Centre for Mathematical Sciences, Lund University, 2005.

[106] P. A. Tresadern and I. D. Reid. Video synchronization from human motion using rank constraints. *Computer Vision and Image Understanding*, 113(8):891–906, Aug. 2009.

[107] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle Adjustment — A Modern Synthesis. In B. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, number 1883 in Lecture Notes in Computer Science, pages 298–372. Springer Berlin Heidelberg, Sept. 1999.

[108] R. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal on Robotics and Automation*, 3(4):323–344, Aug. 1987.

[109] T. Tuytelaars and L. Van Gool. Synchronizing video sequences. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 762–768, 2004.

[110] J. Ventura, C. Arth, and V. Lepetit. An Efficient Minimal Solution for Multi-camera Motion. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 747–755, Los Alamitos, CA, USA, 2015. IEEE Computer Society.

[111] M. Vo, S. G. Narasimhan, and Y. Sheikh. Spatiotemporal Bundle Adjustment for Dynamic 3D Reconstruction. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1710–1718, June 2016.

[112] M. ˇSmíd and J. Matas. Rolling Shutter Camera Synchronization with Sub-millisecond Accuracy. In *Proc. 12th Int. Conf. Comput. Vis. Theory Appl.*, page 8, 2017.

[113] P. Wolf and C. Ghilani. *Adjustment Computations: Statistics and Least Squares in Surveying and GIS*. 3rd Ed). John Wiley & Sons, 1997. LCCB: 96022146.

[114] C. Wu. VisualSFM: A Visual Structure from Motion System, 2011.

[115] Y. Wu and Z. Hu. PnP Problem Revisited. *Journal of Mathematical Imaging and Vision*, 24(1):131–141, Jan. 2006.

[116] C.-K. Yap. *Fundamental Problems of Algorithmic Algebra*, volume 49. Oxford University Press Oxford, 2000.

[117] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, Nov. 2000.

# List of Publications Related to the Thesis

Citations are taken from Google scholar without auto-citations.

## Impacted journal papers excerpted by ISI

C. Albl, Z. Kukelova, V. Larrson, T. Pajdla:
**Rolling Shutter Camera Absolute Pose**
*Transactions on Pattern Analysis and Machine Intelligence*, IEEE, submitted 2018 (1st revision - minor). [60%]

## Conference papers excerpted by ISI

C. Albl, Z. Kukelova, T. Pajdla:
**R6P-Rolling shutter camera absolute pose**
CVPR 2015, Boston, Massachusetts, USA. [50%]
Citations: 30

C. Albl, Z. Kukelova, and T. Pajdla:
**Rolling shutter absolute pose problem with known vertical direction**
CVPR 2016, Las Vegas, Nevada, USA. [50%]
Citations: 9

Z. Kukelova, C. Albl, A. Sugimoto and T. Pajdla:
**Linear solutions to rolling shutter absolute pose**
submitted to ACCV 2018. [30%]
C. Albl, A. Sugimoto, and T. Pajdla:
**Degeneracies in rolling shutter SfM**
ECCV 2016, Amsterdam, Netherlands. [50%]
Citations: 7

C. Albl, Z. Kukelova, A. Fitzgibbon, J. Heller, M. Smid and T. Pajdla:
**On the Two-View Geometry of Unsynchronized Cameras**
CVPR 2017, Honolulu, Hawaii, USA[50%]
Citations: 2

# Publications not related to the thesis

## Conference papers excerpted by ISI

C. Albl and T. Pajdla:
**Global Camera Parameterization for Bundle Adjustment**
VISAPP 2014, Lisbon, Portugal. [60%]
Citations: 2

## Other conference papers

C. Albl and T. Pajdla:
**Constrained Bundle Adjustment for Panoramic Cameras**
CVWW 2013, Hernstein, Austria. [60%]
Citations: 3