DISS. ETH NO. 23315

Immersive Environments: From Virtual Indoor Tours to Outdoor Modelling

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH Zürich

(Dr. sc. ETH Zürich)

presented by

Olivier Saurer

MSc ETH Zurich

born 27.10.1980
citizen of Sigriswil, canton of Bern, Switzerland

accepted on the recommendation of

Examiner
Prof. Dr. Marc Pollefeys

Co-Examiners
Prof. Dr. Friedrich Fraundorfer
Prof. Dr. Davide Scaramuzza
Prof. Dr. Konrad Schindler

2016

Contents

Acknowledgement

The presented thesis would not have been possible without the help and influence of many people. Many had direct impact through scientific collaboration, others had an indirect positive influence on my doctoral studies.

First, I would like to express my great gratitude to my advisor, Prof. Marc Pollefeys, for providing me with the opportunity to pursue my doctoral studies under his supervision. His guidance gave me the opportunity to do research in many great areas and get involved in different exiting projects. These projects gave me the chance to work on different topics and explore different areas of computer vision. I'm very grateful to all my collaborators who help me get through many deadlines and helped laid the foundation of this thesis. Im specially thankful to: Friedrich, Kevin, Gim Hee, Jean-Yves, Pascal, Cedric and Georges. There were many late nights we worked together rushing for paper deadlines.

The Computer Vision and Geometry (CVG) group has been a great environment to peruse work and play. I want to thank Alexander, Amaël, Andrea, Aparna, Bastien, Bernhard, Bo, Christian, Christopher, Dominik, Fabio, Federico, Georges, Ian, Jens, Johannes, Katarina, Kalin, Kevin, Lubor, Luca, Lukas, Lionel, Lorenz, Martin, Nikolay, Pablo, Peidong, Petri, Pranav, Roland, Torsten, Thomas and Yagiz, with whom I had many interesting and fruitful discussions, which led to some of this work. Furthermore I did like to thank Susanne and Thorsten for taking care of all the administrative and technical matters. I'd also like to thank Yean-Yves, Dennis, David, Carlos, Sameer, Changchang and Steve, with whom I spent several summers working on exiting computer vision problems at google.

Finally, and most importantly I'm grateful to my parents and grandparents to give me the possibility to study and pursue a PhD. I'm also grateful to my siblings for their constant help and support.

Abstract

Virtual navigation through urban environments and building interiors have gained great popularity in the last decade. Initially maps were served in topological representation, providing vital information while leaving out unnecessary details. They have since greatly evolved from panoramic images to full 3D representations of the environment.
In this thesis, we propose a suite of algorithms to create such maps. Starting with sparse representation of the 3D environment and creating compelling 3D representation of buildings. Finally we create topological maps and virtual tours through building interiors.

Estimating the camera motion between two intrinsically calibrated cameras, requires a minimal of 5-point correspondences. Given additional scene information, such as the gravity direction and a dominant scene plane (the ground plane for instance), reduces the minimal number of required correspondences. For a known gravity direction, we proposed a minimal 3-point algorithm. This algorithm relies on a known common direction, which is obtained either through IMU measurements or vanishing points. If in addition a dominant plane of the scene is known, such as the ground plane, ceiling or a wall, the minimal number of correspondences required, can further be reduced to 2.5- or 2-points.

Many of today's cameras use a CMOS sensor to capture light and eventually to form an image. The sensor has a peculiarity of sequentially exposing each row/column with light, also known as a rolling shutter. For moving cameras this leads to distortion artefacts. We exploit this time delay in the image formation process to estimate the position and velocity of the camera from a single image, which results in a 5-point absolute pose algorithm.

Applying dense stereo to such imagery, leads to geometrically wrong reconstructions. We propose a novel stereo algorithm that models the temporal delay in the image formation process. The obtained results are geometrically consistent and provide high fidelity to the ground truth.

Indoor reconstructions are known to be challenging. The lack of light often leads to low signal to noise ratio in the image, narrow corridors provide poor visual connectivity between images, which makes the 3D reconstruction process brittle. To overcome these challenges, we propose to build virtual tours from omnidirectional imagery. Such virtual tours provide the user with a great immersive experience, without the need to build a full 3D model.

Zusammenfassung

Virtuelle Navigation durch städtische Umgebungen und Innenräumen haben in über die letzten Jahre an grosser Bedeutung gewonnen. Anfangs wurden Karten in topologischer Form dargestellt um wichtige Informationen hervorzuheben, während unnötige Details ausgelassen wurden. Seitdem haben sich Karten stark weiter entwickelt und reichen von Panorama-Bilder bis hin zur 3D Darstellung der Umgebung.

In dieser Arbeit schlagen wir eine Reihe von Algorithmen vor, um solche Karten zu erstellen. Wir beginnen mit der spärlichen Darstellung der 3D-Umgebung und schaffen dann detaillierte 3D Repräsentationen von Gebäuden. Schliesslich schlagen wir Algorithmen vor um topologische Karten und virtuelle Touren durch Innenräumen zu erstellen.

Die Schätzung der Kamerabewegung zwischen zwei kalibrierten Kameras, benötigt ein Minimum an 5-Punktkorrespondenzen. Mit jeder zusätzlichen Szeneinformation, wie beispielsweise der Schwerkraftrichtung oder einer dominanten Szenen Ebene, kann sich die Anzahl der erforderlichen Punktkorrespondenzen reduzieren. Dazu haben wir einen minimalen 3-Punkt-Algorithmus vorgeschlagen. Dieser Algorithmus beruht auf einer bekannten gemeinsamen Richtung, die entweder durch IMU Messungen oder Fluchtpunkte gewonnen werden kann. Wenn zusätzlich eine dominante Ebene der Szene bekannt ist, wie beispielsweise die Grundebene, Decke oder Wand, kann die Anzahl von Korrespondenzen auf 2,5- oder 2-Punkten reduziert werden.

Viele Kameras die heute verwendet werden, beruhen auf einem CMOS-Sensor. Der Sensor hat die Besonderheit, jede Zeile oder Spalte sequenziell zu belichten, was in der Literatur auch als ”Rolling-Shutter” bekannt ist. Bewegt sich die Kamera, führt diese zeitversetzte Belichtung der einzelnen Zeilen oder Spalten zu Verzerrungen im Bild. Wir nutzen diese Zeitverzögerung im Bilderzeugungsprozess, um die Position und die Geschwindigkeit der Kamera zu schätzen. Die Formulierung resultiert in einem 5-Punkt absolute Pose Algorithmus.

Wendet man auf solchen Rolling-Shutter Bildern Standard Stereo Algorithmen an, entstehen dabei Rekonstruktionen die geometrisch falsch und inkonsistent sind. Wir schlagen deshalb einen neuen Stereo-Algorithmus vor, welcher die zeitlichen Verzögerung im Bilderzeugungsprozess modelliert. Die erzielten Ergebnisse sind geometrisch konsistent und bieten eine hohe Wiedergabetreue zur Ground Truth.

Innen Rekonstruktionen sind bekanntlich schwierig zu realisieren. Der Lichtmangel führt häufig zu geringem Signal-Rausch-Verhältnis im Bild und enge Gänge bieten schlechte visuelle Verbindung zwischen den einzelnen Bildern. Diese Einschränkungen erschweren den 3D-Rekonstruktionsprozess. Aus diesen Gründen schlagen wir vor, virtuelle Touren aus omnidirektionalen Bilder zu realisieren. Eine solche virtuelle Tour bietet dem Anwender eine grosse eindringende Erfahrung, womit die Notwendigkeit ein vollständiges 3D-Modell zu erstellen entfällt.

1 Introduction

Over the last decade the sales of camera phones have over tripled with the evolution of mobile phones. Today the majority of phones have a camera and are also equipped with additional sensors such as Inertial Measurement Unit (IMU). Those sensors allow to measure the gravity direction and the acceleration and orientation of the phone. Using these additional sensor data for camera pose estimation, leads to computationally more efficient algorithms in terms of accuracy and speed.
In addition the majority of todays phones use a CMOS image sensor, which typically have an electronic rolling shutter. Meaning that each image row is exposed sequentially leading to a temporal delay in the image formation process. For a moving camera this delay can lead to severe undesired artefacts in 3D modelling, leading to complete failure of the 3D reconstruction or even worth to geometrically inconsistent models.
We propose a suite of rolling shutter aware algorithms for camera pose estimation and stereo computation.

1.1 Outline of the Thesis

In chapter 2 we introduce the fundamental camera model and algorithms used in this thesis. We discuss the pinhole camera model and the different camera shutter types e.g., the global shutter and the rolling shutter. Furthermore we give an overview of the full 3D reconstruction pipeline from camera pose estimation to dense stereo.

In chapter 3, we explore the different minimal solutions for egomotion estimation the relative pose between two cameras based on a homography formulation with a known gravity directions. These solutions depend on the prior knowledge about the reference plane used by the homography. We then demonstrate that the number of matched points can vary from two to three and that a direct closed-form solution or a Gröbner basis based solution can be derived according to this plane. Many experimental results on synthetic and real sequences in indoor and outdoor environments show the efficiency and robustness of our approach compared to standard methods.

In chapter 4 we look at distortions that are present in images taken from a moving rolling shutter camera. These artefacts degrade the accuracy of absolute camera pose estimation. To alleviate this problem, we introduce an additional linear velocity in the camera projection matrix to approximate the motion of the rolling shutter camera. In particular, we derive a minimal solution using the Gröbner Basis that solves for the absolute pose as well as the motion of a rolling shutter camera. We show that the minimal problem requires 5-point correspondences and gives up to 8 real solutions. We also show that our formulation can be extended to use more than 5-point correspondences. We use RANSAC to robustly obtain all the inliers. In the final step, we relax the linear velocity assumption and do a non-linear refinement on the full motion, i.e., linear and angular velocities, and pose of the rolling shutter camera with all the inliers. We verify the feasibility and accuracy of our algorithm with both simulated and real-world datasets.

Chapter 5 analysis the effect distorted rolling shutter images have on stereo reconstruction. We analyse the case of significant camera motion, e.g. where a bypassing streetlevel capture vehicle uses a rolling shutter camera in a 3D reconstruction framework. The error introduced by the rolling shutter is depth dependent and cannot be compensated based on camera motion/rotation alone, invalidating also rectification for stereo camera systems. On top, significant lens distortion as often present in wide angle cameras intertwines with rolling shutter effects as it changes the *time* at which a certain 3D point is seen. We show that naive 3D reconstructions (assuming global shutter) will deliver biased geometry already for very mild assumptions on vehicle speed and resolution. We then develop rolling shutter dense multiview stereo algorithms that solve for time of exposure and depth at the same time, even in the presence of lens distortion and perform an evaluation on ground truth laser scan models as well as on real street-level data.

Chapter 6 address the problem of sparse to dense 3D reconstruction from rolling shutter images. It is well known that the rolling shutter effect in images captured with a moving rolling shutter camera causes inaccuracies to 3D reconstructions. The problem is further aggravated with weak visual connectivity from wide baseline images captured with a fast moving camera. In this chapter, we propose and implement a pipeline for sparse to dense 3D reconstruction with wide baseline images captured from a fast moving rolling shutter camera. Specifically, we propose a cost function for bundle adjustment that models the rolling shutter effect, incorporates GPS/INS readings, and enforces pairwise smoothness between neighboring poses. We optimize over the 3D structures, camera poses and velocities. We also introduce a novel interpolation scheme for the rolling shutter plane sweep stereo algorithm that allows us to achieve a $7\times$ speed up in the depth map computations for dense reconstruction without losing accuracy. We evaluate our proposed pipeline over a 2.6km image sequence captured with a rolling shutter camera mounted on a moving car.

In chapter 7 we present a semi-automatic method to generate interactive virtual tours from omnidirectional video. Similar to Google Streetview, but focused on indoor environments. The system allows a user to virtually walk through buildings on predefined paths. The user can freely look around and walk into every direction, provided there exists a pre-recorded location, e.g. a bifurcations. The method automatically computes the initial tour topology from the omnidirectional video data using structure from motion. A place recognition step afterwards detects junctions and loop closures. A final interactive refinement step allows to align the initial topology to a floor plan with a few mouse-clicks. The refinement step incorporates both, automatic constraints obtained from the place recognition detection and manual alignment constraints. The presented system combines a high degree of automation with a final user interaction, to create an immersive virtual enviroment.

In chapter 8, we describe a visual localization approach for mobile robots. Robot localization is performed as location recognition. The approach uses global visual features (e.g. GIST) for image similarity and a geometric verification step using vanishing points. Location recognition is an image search to find the most similar image in the database. To deal with partial occlusions, which lower image similarity and lead to ambiguity, vanishing points are used to ensure that a matching database image was taken from the same viewpoint as the query image from the robot. Our approach will assign a query image to a location learned from a training dataset, to an ”Unknown” location or in case of too much uncertainty the algorithm would refrain from a decision. The algorithm was evaluated under the ImageCLEF 2010 RobotVision competition.

Finally chapter 9 recaps the work leading to this dissertation. In addition, we review the contributions and discuss future prospects of indoor and outdoor modelling.

The thesis is based on our findings [89–94], which have been published at various computer vision venues.

2 Foundation

Contents

In this chapter, we discuss the different existing camera models which are used throughout this thesis. We will also discuss the different shutter types, namely the global and rolling shutter which is used in the camera to expose the image sensor to light. We consider two different camera models which are the *Global Shutter Camera Model* and the *Rolling Shutter Camera Model*. In the former case all pixels are exposed at the same time, while in the later case scanlines are exposed sequentially, hence rolling shutter. While both are closely related, the difference becomes apparent when the camera is in motion.

2.1 Pinhole Camera Model

The pinhole camera is a well established camera model in machine vision applications. It represents the projection of a 3D point onto an image sensor. The required transformation can be represented by a 3×3 matrix, usually denoted as $\mathbf{K}$:

$$\mathbf{K} = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.1}$$

The camera's focal length is denoted by f_x and f_y. If the aspect ratio of the sensor's pixel is squared, then the aspect ratio is 1 and therefore $f_x = f_y$. If f_x/f_y is not equal to one, this means that the sensor pixel is not squared. The skew, represented as s, allows for non rectangular pixels. In general pixels are rectangular and therefore $s = 0$.

So far we have considered a camera, where the lens distortion artefacts have been removed from the image. To be able to radially undistorted an image, the lens distortion parameters need to be estimated.

Different camera lens models exist, the choice of the lens distortion parametrization depends on the physical lens. In our work we make use of lenses with little distortion, which satisfy the distortion model proposed by Brown [17]. The model consists of 3 distortion parameters and 2 tangential parameters. Let $\mathbf{X} = [X, Y, Z]^\top$ be an arbitrary 3D point. The projection of the 3D point $\mathbf{X}$ into the camera yields the normalized image coordinates $\mathbf{x}_n$ given as:

$$\mathbf{x}_n = \begin{bmatrix} X/Z \\ Y/Z \end{bmatrix} \tag{2.2}$$

Applying the distortion parameters to the normalized image coordinates gives the distorted image point:

$$\mathbf{x}_d = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6)\mathbf{x}_n + \mathbf{dx}, \tag{2.3}$$

$$\mathbf{dx} = \begin{pmatrix} 2k_4 xy + k_5(r^2 + 2x^2) \\ k_4(r^2 + 2y^2) + 2k_5 xy \end{pmatrix}. \tag{2.4}$$

The coefficients $[k_1, k_2, k_3]$ denote the radial distortion coefficient, $[k_4, k_5]$ the tangential distortion parameters and $r^2 = \sqrt{x^2 + y^2}$.

Given the distortion parameters, we can computed an inverse lookup table, to undistort each pixel in the image.

2.2 Camera Shutter Model

The time a pixel is exposed to light is controlled by the shutter of the camera. The shutter, when open allows light to fall onto the image sensor where photons get integrated. When closed, the shutter barres photons from falling onto the sensor and therefore no photons get integrated. Assuming a moving camera or a moving/changing scene the way the shutter operates can have sever effects on the image formation. The following two subsections discuss the global- and rolling-shutter model.

2.2.1 Global Shutter

A global shutter allows to expose all pixels simultaneously to light. Once the desired exposure time has expired all pixels are simultaneously shielded from the light. This functionality is provided by the majority of CCD sensors. The ability of simultaneously controlling all pixels allows to capture a true snapshot of the scene. An exposure pattern of the CCD sensor is given in Fig. 2.1.

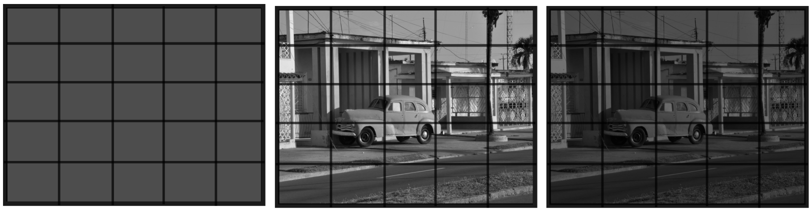


Figure 2.1: Exposure pattern of a global shutter camera. From left to right: First the shutter is closed, then opens up fully to expose all pixels simultaneously. Then the shutter closes again.

2.2.2 Rolling Shutter

Unfortunately the majority of todays sensors are CMOS sensors which are inherently rolling shutter sensors. Sometimes the sensors are combined with mechanical shutters, which can be used for much lower time delays, which in return reduces the shutter scan time and with it the distortion artefacts. The shutter slit slides over the sensor exposing each scanline sequentially, which can lead to artefacts. A sample distorted image is given in Fig. 2.2. In this example the shutter moved from bottom to top while the car was moving from left to right, which makes the scene appear slanted.

To explain why the scene is distorted it is easier to assume a static scene with a moving camera. While the shutter scans over the sensor the scanlines (image rows or columns) are exposed simultaneously at a certain camera position. Due to the moving camera and the sequential exposure of the sensors, each scanline is exposed at a different spatial location, leading to the final distorted image. When the exposure time is far smaller than the scan time of the sensor, this results in a slit sliding over the sensor and letting light fall onto the sensor, this exposure pattern is illustrated in Fig. 2.2.

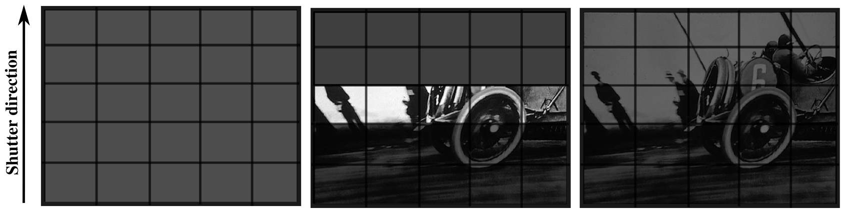


Figure 2.2: Exposure pattern of a rolling shutter camera. First shutter is closed, then a slit slides over the sensor, exposing each image row sequentially. In this example the mechanical shutter moves from bottom to top. Image courtesy of Jacques-Henri Lartigue.

The sequential exposure can lead to different, almost artistic, images which can break standard 3D vision algorithms. Fig. 2.3(a) shows an image of a rotating propeller captured by a static rolling shutter camera. Fig. 2.3(b), shows a scene captured from a moving rolling shutter camera, resulting in slanted vertical structures.

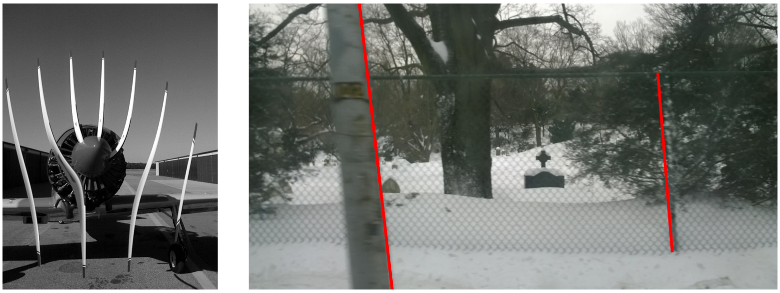


Figure 2.3: Images captured with a rolling shutter camera. (a) Rotating propeller captured with a static rolling shutter camera (courtesy of Soren Ragsdale). (b) Moving rolling shutter camera, capturing a static scene. Due to the camera motion, vertical structures appear slanted in the image.

2.3 Camera Motion Model

For global shutter cameras the pose of the camera is described by the camera orientation $\omega \in SO3$ and a camera position vector $\mathbf{C} \in \mathbb{R}^3$. A arbitrary 3D point $\mathbf{X}_w \in \mathbb{R}^3$ is then transformed into the camera coordinate frame $\mathbf{X}_c \in \mathbb{R}^3$ using:

$$\mathbf{X}_c = \exp(\omega)(\mathbf{X}_w - \mathbf{C}). \tag{2.5}$$

For the more general case of a moving rolling shutter camera a different formulation is required, which models the camera motion during the scan time of the sensor. As proposed in [37] different time continuous camera pose parametrization have been proposed, which lead to the more general formulation:

$$\mathbf{X}_{c,\tau} = \exp(\omega(\tau))(\mathbf{X}_w - \mathbf{C}(\tau)). \tag{2.6}$$

The time dependent orientation $\omega(\tau)$ and pose $\mathbf{C}(\tau)$ can be chosen arbitrarily, depending on the capturing scenario. For a car mounted camera a different motion model would be used than for a hand-held camera. In the former case the dominate motion is the translation motion, while for the latter the dominant motion is the rotation, coming from hand jittering. We further discuss different motion models in chapter 4 and chapter 5.

2.4 Structure-From-Motion

Given a set of images, the task of structure-from-motion is to compute the camera extrinsic parameters (sometime also the intrinsics are estimated at the same time) and simultaneously estimate a sparse representation of the scene. A classical structure-from-motion pipeline is drawn in Fig. 2.4. In the following we give an overview of the full reconstruction process.

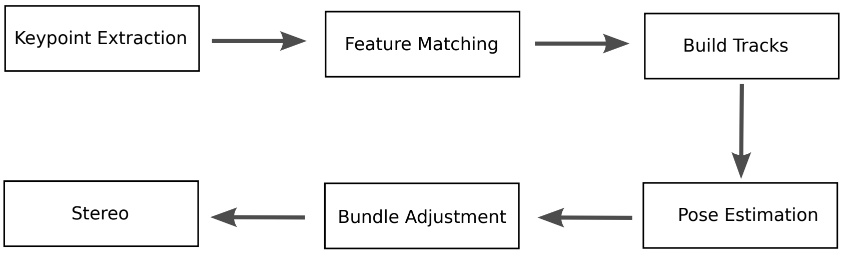


Figure 2.4: Structure-from-Motion pipeline overview.

In a first stage after images are captured, keypoints are extracted from the images. Throughout this work we used SIFT [68] as features. In the next stage matching image pairs are found by matching SIFT feature descriptors followed by a geometric verification. Keypoints which matched over multiple images, form 2D tracks. A track is represented by a 3D point which is visible over multiple images. To bootstrap the reconstruction process, normally two or three views are used to estimate the initial relative pose between the camera, using the essential or fundamental matrix [44]. Then additional views are added to the seed by either expanding the existing reconstruction using relative pose algorithms or by estimating the absolute pose of the image using [74].

Either of the two steps is typically followed by a non-linear refinement of the model, called bundle adjustment. Given a set of camera parameters and a set of tracks, bundle adjustment optimizes the following non-linear least squares error:

$$\sum_i \sum_j ||\mathbf{P}_i \mathbf{X}_j - x_{ij}||^2 \tag{2.7}$$

where $\mathbf{P}_i$ denotes the i^{th} camera projection matrix, $\mathbf{X}_j$ the 3D point and x_ij the keypoint detected in image i, representing the 3D point j.

Recently many SfM algorithms also use unordered image sets, captured at different time instances [1,28]. The main challenges of using unordered image sets is to find the corresponding matching images. Recent development of fast and high quality feature detectors and descriptors have enabled reconstructions from a large set of unordered images, consisting of over 100M images [46].

2.5 Dense Reconstruction

Many different information cues exist to compute the 3D geometry from a set of images. The most widely known are: depth form defocus, shape from shading, voxel carving and stereo correspondence. In terms of robustness stereo based approaches have been shown to be the most successful. Different types of algorithm exist, such as Patch-Match [11, 35], PMVS [33], Plane Sweep [19]. Those algorithms are typically used in a multi-view configuration, meaning that more than two views are used to estimate the scene depth. This makes the depth estimation more robust, especially when dealing with partial occlusions. All three approaches rely on the same input, which is a set of fully calibrated images. Meaning the camera's intrinsics and extrinsic parameters are known.

In this section we provide an overview of the widely used plane-sweep stereo algorithm, which will also form the basis to chapter 5. The algorithm gained large popularity due to it's simple structure which can be implemented efficient on the graphics processing unit (GPU) [19, 36, 115].

The algorithm sweeps through a family of parallel planes, by warping an image from a target view into a reference view, using a plane induced homography. Other more GPU friendly approaches exist [115], which warp the captured image onto a common plane in 3D. Then a photo-consistency value is computed for each plane. For each pixel the depth value of the plane is chosen, which provides the best photo-consistency cost (winner-takes-all). Most algorithm assume the surface is viewed fronto-parallel by the reference view and therefore sweep in a fronto-parallel direction. It has been shown in [36] that when the plane orientation does not align with the scene structure, the matching windows do not perfectly align and therefore provide a lower matching score, than when the plane aligns with the scene structure. Gallup et al. propose in [36] to use multiple sweeping direction, which align with the scene structure. In practise this provides better photo-consistency when the 3D point is hypothesized at the right depth. The algorithm provides one depthmap for each considered direction, the final depthmap is then obtained by fusing the different depthmaps. An illustration of the plane-sweep algorithm is given in Fig 2.5.

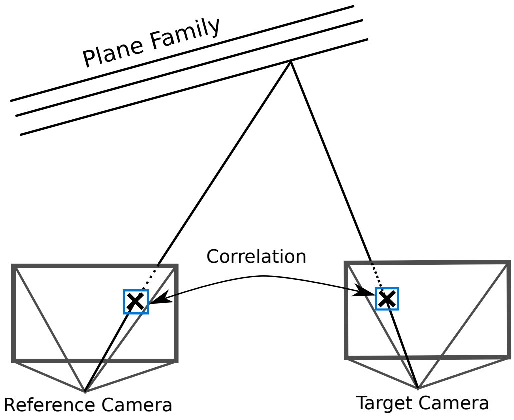


Figure 2.5: Plane-sweep stereo, pixel transfer over a hypothesised plane, from target camera into the reference camera.

3 Homography Based Camera Pose Estimation with a Known Gravity Direction

Contents

Nowadays, point-based methods to estimate the motion of a camera are well known. If the camera is uncalibrated, eight or seven points are needed to estimate the fundamental matrix between two consecutive views [44]. When the intrinsic parameters of the camera are known, five points are then enough to estimate the essential matrix [78]. To decrease the sensitivity of these methods, a robust framework such as Random Sample Consensus (RANSAC) is necessary. Thus, reducing the number of needed matched points between views is important in terms of computation efficiency and of robustness improvement. For example, as shown in Figure 3.1, for a probability of success of 0.99 and a rate of outliers equal to 0.5, the number of RANSAC trials is divided by eight, if five points are used instead of eight. In the case of a robust estimation based on eight points, 1177 trials are necessary whereas 145 are sufficient if only five points are required. Thus, finding a minimal solution for egomotion estimation is important for robust real time applications.

However, reducing the number of necessary points is only possible if some hypotheses or supplementary data are available. For example, if we know a common direction between the two views, three points can then be used to estimate the full essential matrix [29]. Extreme situations appear when a planar non-holonomic motion is supposed [95] or when the metric velocity of a single camera can be estimated knowing its attitude and its acceleration [52]. In these cases, only one point allows to estimate the motion. These initial hypotheses or additional knowledges can then deal with the pose of the camera or with the 3D structure of the scene. For example, if the 3D points belong to a single plane, the egomotion estimation is reduced to a homography computation between two views, that can be calculated using only four points [44]. In many scenes and many applications, the scene plane hypothesis seems suitable. Indeed, in many scenarios such as indoor or street corridors and more generally in man made environments, this assumption holds.

Thus, in this chapter we investigate the cases where at least one plane is present in the scene and where we have some partial knowledge about the pose of the camera. We suppose that we are able to extract a common direction between consecutive views and we can have some information about the normal of the considered plane. Obtaining a common direction can be easily performed thanks to an IMU (Inertial Measurement Unit) associated with the camera, which is often the case in mobile devices or UAV (Unmanned Aerial Vehicle). The coupling with a camera is then very easy and can then be used for different computer vision tasks [20,24,65,113,114]. Without any external sensor, this common direction can also be directly extracted from the images thanks to vanishing points [14] or horizon detection [81].

In this work, assuming the roll and pitch angles of the camera as known, we propose to find a minimal closed-form solution for homography estimation in man made environments. We will derive different solutions depending on the prior knowledge about the 3D scene:

- If the extracted points lie on the ground plane, we will see that only two points are required to estimate the camera egomotion. In this case, the solution is unique and contrary to the other algorithms for essential

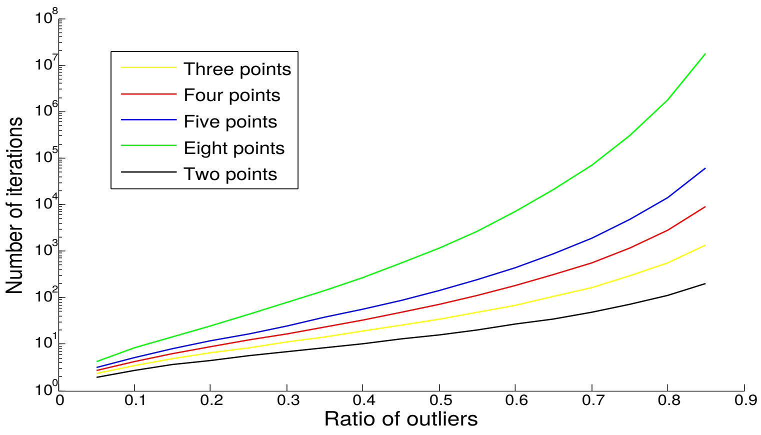


Figure 3.1: Comparison of the RANSAC iteration number for 99% of success probability

matrix estimation, there is no supplementary verification for finding the good solutions among the different possibilities.

- If the considered points are on a vertical plane, we propose an efficient 2.5pt formulation in order to retrieve the motion of the camera and the normal of the plane related to the pose of the camera. This solution allows for an early reject of a pose hypothesis by including a consistency check on the three point correspondences.

- If the plane orientation is completely unknown, we develop a minimal solution using only three points instead of four points needed in the classical homography estimation.

All these methods will be evaluated on synthetic and real data and compared with different methods proposed in the literature.
The rest of the chapter is organized as follows. In the second part, we describe the different existing methods in the literature which deal with minimal solution for egomotion estimation. In the next section, we explain how to reduce

the number of points for estimating the homography between two views and derive the proposed solutions according to the prior knowledge. In the fourth section, we show the behaviour of our solutions on synthetic and real data and compare with other classical methods in a quantitative evaluation. Finally, we will conclude by providing some extents to this work.

3.1 Related Work

When the camera is not calibrated, at least 8 or 7 points are needed to recover the motion between views [44]. It's well known, that if the camera is calibrated, only 5 feature point correspondences are sufficient to estimate the relative camera pose. Reducing this number of points can be very interesting in order to reduce the computation time and to increase the robustness when using a robust estimator such as RANSAC. The reduction of the degree of freedom (DoF) number and consequently the number of matched points between images can be achieved by introducing some constraints on the camera motion (planar for example) or the feature points (on the same plane) or by using some additional information provided by other sensors such as IMU for instance.

For example, if all the 3D points lie on a plane, a minimum of 4 points is required to estimate the motion of the camera between two-views [44]. On the other hand, if the camera is embedded on a mobile robot which moves on a planar surface, only 2 points are required to recover the motion [82] and if in addition the mobile robot has non-holonomic constraints only one point is necessary [95]. Similarly, if the camera moves in a plane perpendicular to the gravity, 1 point correspondence is sufficient to recover the motion as shown by Troiani et al. [108].

The number of points needed to estimate the egomotion can be also reduced if some information about the relative rotation between two poses are available. This information can be given by vanishing points extraction in the images [13] or by taking into account extra information given by an additional sensor. Thus, Li et al. [60] show that in the case of an IMU associated to the camera, only 4 points are sufficient to estimate the relative motion even if the extrinsic calibration between the IMU and the camera is not known.

Similarly, some different algorithms have been recently proposed in order to estimate the relative pose between two cameras by knowing a common direction. It has been demonstrated that knowing roll and pitch angles of the camera at each frame, only three points are needed to recover the yaw angle and the translation of the camera motion up to scale [29, 49, 76]. In these approaches, only the formulation of the problem is different and consequently the way to solve it. All these works start with a simplified essential matrix in order to derive a polynomial equation system. For example, in [49], their parametrization leads to 12 solutions by using the Macaulay matrix method. The correct solution has then to be found among a set of possible solutions. The approach presented in [29] permits to obtain a 4^{th}-order polynomial equation and consequently leads to a more efficient solution. In [76], the authors propose a closed-form solution to this 4^{th}-order polynomial equation that allows a faster computation.

For a further reduction of necessary feature points, stronger hypotheses have to be added. If the complete rotation between the two views are known, only 2 degrees of freedom corresponding to the translation up-to-scale has to be estimated and consequently 2 points are sufficient to solve the problem [12, 109]. In this case, the authors compute the translation vector using the epipolar geometry given the rotation. Thus, these approaches allow to reduce the number of points but also imply the knowledge of the complete rotation between two views making the pose estimation very sensitive to IMU inaccuracy. More recently, Martinelli [72] proposes a closed-form solution for structure from motion knowing the gravity axis of the camera in a multiple view scheme. He shows that at least three feature points lying on a same plane and three consecutive views are required to estimate the motion. In the same way, the plane constraint has been used for reducing the complexity of the bundle adjustment (BA) in a visual simultaneous localization and mapping (SLAM) embedded on a micro-aerial vehicle (MAV) [55].

Most closely related papers to our approach are the works of [29, 76] in which they simplify the essential matrix knowing the vertical of the cameras. In this work, to reduce the number of points, rather than deriving the epipolar constraint to compute the essential matrix, we propose to use the homography constraint between two views. Thus, we suppose that a significant plane exists in the scene and that the gravity direction is known. Let us note that recently, in [107] Troiani et al. have also proposed a method using 2 points on the ground plane with the knowledge of the vertical of the camera. However, they do not use the homography formalism and their method requires to know the distance between the two 3D points. In our method, this hypothesis is not necessary and we only assume that the points lie on a same plane. The Manhattan world assumption [21] has also recently successfully been used for multi-view stereo [31], the reconstruction of building interiors [32] and also for scene reconstruction from a single image only [54]. Our contribution differs from them, as we combine gravity measurements with the weak Manhattan world assumption. This chapter is an extension of [91, 94] where we studied camera pose estimation based on homographies with a common vertical direction and a known or at least partially known plane normal. In [91] we proposed a homography based pose estimation algorithm that does not require any knowledge on the plane normal. In fact the algorithm provides the plane normal in addition to the camera pose.

3.2 Relative Pose Algorithms

Knowing the vertical direction in images will simplify the estimation of camera pose and camera motion, which are fundamental methods in 3D computer vision. It is then possible to align every camera coordinate system with the measured vertical direction such that the z-axis of the camera is parallel to the vertical direction and the x-y-plane of the camera is orthogonal to the vertical direction (illustrated in Fig. 3.2). In addition, this would mean that the x-y-plane of the camera is now parallel to the world's ground plane and the z-axis is parallel to vertical walls.

This alignment can just be done as a coordinate transform for motion estimation algorithms, but also be implemented as image warping such that feature extraction methods benefit from it. Relative motion between two such aligned cameras reduces to a 3-DOF motion, which consists of 1 remaining rotation and a 2-DOF translation vector (i.e., a 3D translation vector up to scale).

The algorithms for estimating the relative pose are derived from a homogra-

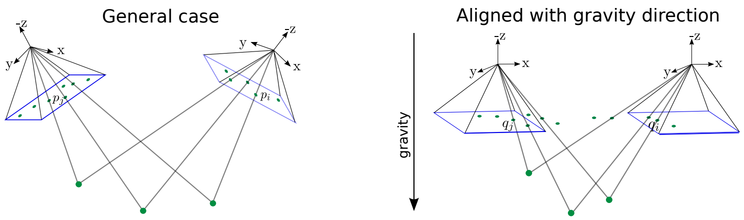


Figure 3.2: Alignment of the camera with the gravity direction.

phy formulation, where a plane is observed in two images. The homography is then decomposed into a relative rotation and translation between the two images. By incorporating the known vertical direction and allowing for strictly vertical or horizontal planes, the parametrization of the pose estimation problem is greatly reduced from 5-DOF to 3-DOF. This simplification leads to a closed-form 2pt and a 2.5pt algorithm to compute the homography. By relaxing the assumption of strictly vertical or horizontal structures and making use of the known gravity direction, the homography formulation results in a closed form solution requiring 3-points only.

In the following subsections we derive the 2pt algorithm for the known plane normal cases (ground and vertical plane), then we provide a derivation of the 2.5pt and 3pt algorithm for a known gravity direction with an unknown plane orientation.

3.2.1 2pt Relative Pose for Points on the Ground Plane

The general homographic relation for points belonging to a 3D plane and projected in two different views is defined as follows:

$$\mathbf{q}_j = \mathbf{H}\mathbf{q}_i, \tag{3.1}$$

with $\mathbf{q}_i = [x_i, y_i, w_i]^\top$ and $\mathbf{q}_j = [x_j, y_j, w_j]^\top$ the projective coordinates of the points between the views i and j. $\mathbf{H}$ is given by:

$$\mathbf{H} = \mathbf{R} - \frac{1}{d}\mathbf{t}\mathbf{n}^\top, \tag{3.2}$$

where $\mathbf{R}$ and $\mathbf{t}$ are respectively the rotation and the translation between views i and j and where d is the distance between the camera i and the 3D plane described by the normal $\mathbf{n}$.
In our case, we assume that the camera intrinsic parameters are known and that the points $\mathbf{q}_i$ and $\mathbf{q}_j$ are normalized. We also consider that the attitude of the cameras for the both views are known and that these attitude measurements have been used to align the camera coordinate system with the ground plane. In this way, only the yaw angle θ between the two views remains unknown. Similarly, since we consider that the ground plane constitutes the visible 3D plane during the movement of the cameras, we can note that $\mathbf{n} = [0, 0, 1]^\top$. Consequently, equation(3.2) can be written as:

$$\mathbf{H} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} - \frac{\mathbf{t}}{d} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}^\top, \tag{3.3}$$

d being unknown, translation can be known only up to scale. Consequently, the camera-plane distance d is set to 1 and absorbed by $\mathbf{t}$. We then obtain:

$$\begin{aligned} \mathbf{H} &= \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}^\top, & (3.4) \\ &= \begin{bmatrix} \cos(\theta) & -\sin(\theta) & -t_x \\ \sin(\theta) & \cos(\theta) & -t_y \\ 0 & 0 & 1 - t_z \end{bmatrix}. & (3.5) \end{aligned}$$

In a general manner, this homography can be parametrized as

$$\mathbf{H} = \begin{bmatrix} h_1 & -h_2 & h_3 \\ h_2 & h_1 & h_4 \\ 0 & 0 & h_5 \end{bmatrix}. \tag{3.6}$$

The problem consists of solving for the five entries of the homography $\mathbf{H}$. We consider the following relation:

$$\mathbf{q}_j \times \mathbf{H}\mathbf{q}_i = \mathbf{0}, \tag{3.7}$$

where $\times$ denotes the cross product. By rewriting the equation, we obtain:

$$\begin{bmatrix} x_j \\ y_j \\ w_j \end{bmatrix} \times \begin{bmatrix} h_1 & -h_2 & h_3 \\ h_2 & h_1 & h_4 \\ 0 & 0 & h_5 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ w_i \end{bmatrix} = \mathbf{0}. \tag{3.8}$$

This gives us three equations, where two of them are linearly independent. We expand the above equation and consider only the first two linearly independent equations, which results in:

$$\begin{bmatrix} -w_j y_i h_1 - w_j x_i h_2 - w_i w_j h_4 + w_i y_j h_5 \\ w_j x_i h_1 - w_j y_i h_2 + w_i w_j h_3 - w_i x_j h_5 \end{bmatrix} = \mathbf{0}. \tag{3.9}$$

The equation system can be re-written into:

$$\begin{bmatrix} -w_j y_i & -w_j x_i & 0 & -w_i w_j & w_i y_j \\ w_j x_i & -w_j y_i & w_i w_j & 0 & -w_i x_j \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \end{bmatrix} = \mathbf{0}. \tag{3.10}$$

The above equation represents a system of equation of the form $\mathbf{A}\mathbf{h} = 0$. It is important to note that $\mathbf{A}$ has rank 4. Since each point correspondence gives rise to two independent equations, we require two point correspondences to solve for $\mathbf{h}$ up to one unknown scale factor. The singular vector of $\mathbf{A}$, which has the smallest singular value spans a one dimensional (up to scale) solution space. We chose the solution $\mathbf{h}$ such that $||\mathbf{h}|| = 1$. Then, to obtain valid rotation parameters we enforce the trigonometric constraint $h_1^2 + h_2^2 = 1$ on $\mathbf{h}$, by dividing the solution vector by $\pm\sqrt{h_1^2 + h_2^2}$. The camera motion parameters, can directly be derived from the homography:

$$\mathbf{t} = \begin{bmatrix} -h_3, & -h_4, & 1-h_5 \end{bmatrix}^\top, \tag{3.11}$$

$$\mathbf{R} = \begin{bmatrix} h_1 & -h_2 & 0 \\ h_2 & h_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.12}$$

Due to the sign ambiguity in $\pm\sqrt{h_1^2 + h_2^2}$ we obtain two possible solutions for $\mathbf{R}$ and $\mathbf{t}$.

3.2.2 2pt Relative Pose for a Known Vertical Plane Normal

The following algorithm is able to compute the relative pose given 2 point correspondences and the normal of the plane on which the points reside. The derivation will be carried out for a vertical plane but works similar for planes parametrized around other axis.

The homography for a vertical plane can be written as:

$$\mathbf{H} = \mathbf{R_z} - [t_x, t_y, t_z]^\top [n_x, n_y, 0], \tag{3.13}$$

where $\mathbf{R_z}$ denotes the rotation matrix around the z-axis.

Expanding the expression in (3.13) we obtain:

$$\mathbf{H} = \begin{bmatrix} \cos(\theta) - n_x t_x & -\sin(\theta) - n_y t_x & 0 \\ \sin(\theta) - n_x t_y & \cos(\theta) - n_y t_y & 0 \\ -n_x t_z & -n_y t_z & 1 \end{bmatrix}, \tag{3.14}$$

$$= \begin{bmatrix} h_1 & h_2 & 0 \\ h_3 & h_4 & 0 \\ h_5 & \frac{n_y}{n_x} h_5 & 1 \end{bmatrix}. \tag{3.15}$$

This leaves 5 entries in $\mathbf{H}$ to be estimated. Each point correspondence gives 2 inhomogeneous linearly independent equations of the form $\mathbf{Ah} = \mathbf{b}$. Using equation 3.7 we obtain:

$$\begin{bmatrix} -d - h_3 a - h_4 b + h_5 x_i y_j + h_5 y_i c \\ -e + h_1 a + h_2 b - h_5 x_i x_j - h_5 x_j c \end{bmatrix} = \mathbf{0}, \tag{3.16}$$

$$\begin{bmatrix} 0 & 0 & -a & -b & x_i y_j + y_i c \\ a & b & 0 & 0 & -x_i x_j - x_j c \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \end{bmatrix} = \begin{bmatrix} d \\ e \end{bmatrix}, \tag{3.17}$$

with:

$$a = w_j x_i,\ b = w_j y_i,\ c = y_j \frac{n_y}{n_x},\ d = -w_i y_j,\ e = w_i x_j.$$

Using 2 point correspondences, this gives 4 equations which is a deficient-rank system. The solution is $\mathbf{h} = \mathbf{V}\mathbf{y} + w\mathbf{v}$ (see [44]) where $svd(\mathbf{A}) = \mathbf{U}\mathbf{D}\mathbf{V}^\top$ and $\mathbf{v}$ is the last column vector of $\mathbf{V}$. The vector $\mathbf{y}$ is computed by $\mathbf{y_i} = \mathbf{b'_i}/d_i$ where d_i is the i^{th} diagonal entry of $\mathbf{D}$ and $\mathbf{b}' = \mathbf{U}^\top \mathbf{b}$.

This leaves the unknown scalar w which can be computed from an additional constraint. For this particular problem the use of the trigonometric constraint ($\cos(\theta)^2 + \sin(\theta)^2 - 1 = 0$) is preferred over the determinant constraint ($\det(\mathbf{H}^\top \mathbf{H} - \mathbf{I}) = 0$).

The trigonometric constraint can be fully expressed in terms of the variables h_1, h_2, h_3, h_4.

$$\cos(\theta)^2 + \sin(\theta)^2 - 1 = 0 \tag{3.18}$$

$$(h_1 + n_x t_x)^2 + (-h_2 - n_y t_x)^2 - 1 = 0 \tag{3.19}$$

with:

$$t_x = n_x(h_4 - h_1) - n_y(h_2 + h_3) \tag{3.20}$$

Substituting symbolically the entries of $\mathbf{h} = \mathbf{V}\mathbf{y} + w\mathbf{v}$ into Eq. 3.19 results in a quadratic equation in the remaining unknown w (the expanded equation is not shown due to its excessive length). Solving for the variable w gives two solutions for the parameters h_1, h_2, h_3, h_4, h_5.

Once the homography $\mathbf{H}$ is estimated it can be decomposed into relative rotation and relative translation parameters. We back-substitute the entries of $\mathbf{H}$ from (3.15), that is h_1, h_2, h_3, h_4 and h_5 into (3.14). Knowing n_x and n_y, the translation parameters can directly be computed using the following relations:

$$\begin{aligned} t_z &= \frac{-h_5}{n_x}, && (3.21)\\ t_x &= n_x(h_4 - h_1) - n_y(h_2 + h_3), && (3.22)\\ t_y &= n_y(h_1 - h_4) - n_x(h_2 + h_3). && (3.23) \end{aligned}$$

And the rotation parameter is then obtained through:

$$\tan\theta = \frac{h_3 + n_x t_y}{h_1 + n_x t_x}. \qquad (3.24)$$

3.2.3 2.5pt Relative Pose with Unknown Vertical Plane Normal

The 2.5pt algorithm is an extension of the 2pt algorithm described in section 3.2.2. The homography is parametrized as in (3.13). However, when the plane normal $\mathbf{n}$ is not known we can't make use of the same linear constraint, thus all the 6 parameters of $\mathbf{H}$ have to be estimated. To do so, one more equation is required which can be taken from a third point. Thus we stack the constraint equations of 2 points and 1 of the equations from a third point into an equation system of the form $\mathbf{Ah} = \mathbf{b}$. For one point correspondence two equations can be derived as follows. First the homography is defined as:

$$\mathbf{H} = \begin{bmatrix} \cos(\theta) - n_x t_x & -\sin(\theta) - n_y t_x & 0 \\ \sin(\theta) - n_x t_y & \cos(\theta) - n_y t_y & 0 \\ -n_x t_z & -n_y t_z & 1 \end{bmatrix}, \tag{3.25}$$

$$= \begin{bmatrix} h_1 & h_2 & 0 \\ h_3 & h_4 & 0 \\ h_5 & h_6 & 1 \end{bmatrix}. \tag{3.26}$$

The computing $\mathbf{q}_j \times \mathbf{H}\mathbf{q}_i$ leads to:

$$\begin{bmatrix} -c - h_3 a - h_4 b + h_5 x_i y_j + h_6 y_i y_j \\ -d + h_1 a + h_2 b - h_5 x_i x_j - h_6 x_j y_i \end{bmatrix} = \mathbf{0}, \tag{3.27}$$

$$\begin{bmatrix} 0 & 0 & -a & -b & x_i y_j & y_i y_j \\ a & b & 0 & 0 & -x_i x_j & -x_j y_i \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \end{bmatrix} = \begin{bmatrix} c \\ d \end{bmatrix}, \tag{3.28}$$

with:

$$a = w_j x_i, \; b = w_j y_i, \; c = -w_i y_j, \; d = w_i x_j.$$

As in section 3.2.2 the solution to this system is of the form $\mathbf{h} = \mathbf{V}\mathbf{y} + w\mathbf{v}$. The unknown scalar w can be computed from the additional homography constraint $\det(\mathbf{H}^\top \mathbf{H} - \mathbf{I}) = 0$, see [71]. The determinant is a 4^{th} order polynomial in w which results in 4 solutions for $\mathbf{H}$.

The decomposition of the homography into translation and rotation parameters of the relative motion follows the same steps as the one in section 3.2.2. However, it differs as the normals n_x and n_y are not given and need to be computed in the process. We again back-substitute the entries of $\mathbf{H}$ from (3.26) into (3.25). First we compute t_z using the relation $n_x^2 + n_y^2 = 1$,

$$t_z = \pm\sqrt{h_5^2 + h_6^2}. \tag{3.29}$$

This gives two solutions for t_z which differ in the sign. Now the unknown normals can be computed.

$$n_x = \frac{-h_5}{t_z}, \tag{3.30}$$

$$n_y = \frac{-h_6}{t_z}. \tag{3.31}$$

After this we can follow again the procedure of section 3.2.2 and compute the remaining parameters with the following equations, however, using both solutions for t_z, n_x and n_y,

$$t_x = n_x(h_4 - h_1) - n_y(h_2 + h_3), \tag{3.32}$$

$$t_y = n_y(h_1 - h_4) - n_x(h_2 + h_3), \tag{3.33}$$

$$\tan\theta = \frac{h_3 + n_x t_y}{h_1 + n_x t_x}. \tag{3.34}$$

The interesting fact in this case is, that we only use one of the two available equations from the third point. While in the RANSAC loop we still need to sample 3 points for this method, it is now possible to do a consistency check on the 3 point correspondences. To be an outlier free homography hypothesis the one remaining equation has also to be fulfilled. This can easily be tested and if it is not fulfilled the hypothesis is prematurely rejected. This gives a computational advantage over the standard 3pt essential matrix method [29], because inconsistent samples can be detected without testing on all the other point correspondences.

3.2.4 3pt Relative Pose using the Homography Constraint

In this section we discuss a 3pt formulation of the camera pose estimation with a known vertical direction. It differs from the algorithms in the previous section as it does not need the presence of scene planes. A 3pt algorithm has already been presented by [29] but using an essential matrix formulation. With this 3pt algorithm we propose an alternative to the previous essential matrix algorithm but based on a homography formulation. We start from (3.13), instead of assuming the plane to be parallel to the gravity vector we don't make any assumption on the plane orientation and therefore use 3 parameters n_x, n_y, n_z, for the fully unknown plane normal, which leads to:

$$\mathbf{H} = \mathbf{R_z} - [t_x, t_y, t_z]^\top [n_x, n_y, n_z]. \tag{3.35}$$

The camera-plane distance is absorbed by **t** the same way as in the previous sections.

The homography matrix then consists of the following entries:

$$\mathbf{H} = \begin{bmatrix} \cos(\theta) - t_x n_x & -\sin(\theta) - t_x n_y & -t_x n_z \\ \sin(\theta) - t_y n_x & \cos(\theta) - t_y n_y & -t_y n_z \\ -t_z n_x & -t_z n_y & 1 - t_z n_z \end{bmatrix}. \tag{3.36}$$

The unknowns we are seeking for are the motion parameters $\cos(\theta)$, $\sin(\theta)$, t_x, t_y, t_z and the normal $[n_x, n_y, n_z]$ of the plane spanned by the 3 point correspondences. Recall that the standard 3pt essential matrix algorithm only solves for the camera motion, while the 3pt homography algorithm provides the camera motion and a plane normal with the same number of correspondences. To solve for the unknowns we setup an equation system of the form: $\mathbf{q}_j \times \mathbf{H}\mathbf{q}_i = 0$ and expand the relations to obtain the following two polynomial equations:

$$a t_y - b t_z - w_j x_i \sin(\theta) - w_j y_i \cos(\theta) + y_j w_i = 0, \tag{3.37}$$

$$-a t_x + c t_z + w_j x_i \cos(\theta) - w_j y_i \sin(\theta) - x_j w_i = 0, \tag{3.38}$$

where:

$$\begin{aligned} a &= w_j x_i n_x + w_j y_i n_y + w_j n_z w_i, \\ b &= y_j w_i n_z + y_j n_x x_i + y_j n_y y_i, \\ c &= x_j n_x x_i + x_j w_i n_z + x_j n_y y_i. \end{aligned} \tag{3.39}$$

The third equation obtained from $\mathbf{q}_j \times \mathbf{H}\mathbf{q}_i = \mathbf{0}$ is omitted since it is a linear combination of the two other equations. Therefore each point correspondence gives 2 linearly independent equations and there are two additional quadratic constraints, the trigonometric constraint and the unit length of the normal vector that can be utilized:

$$\sin^2(\theta) + \cos^2(\theta) = 1, \tag{3.40}$$

$$n_x^2 + n_y^2 + n_z^2 = 1. \tag{3.41}$$

The total number of unknowns is 8 and the two quadratic constraints together with the equations from 3 point correspondences give a total of 8 polynomial equations in the unknowns. An established way to find an algebraic solution to such a polynomial equation system is by using the Gröbner basis technique [22]. By computing the Gröbner basis a univariate polynomial can be found, which allows to find the value of this variable by root solving. The remaining variables can then be computed by back-substitution. To solve our problem we use the automatic Gröbner basis solver by Kukelova et al. [53], which can be downloaded at the authors webpage. The software automatically generates Matlab-Code that computes a solution to the given polynomial equation system (in our case the above specified 8 equations). The produced Matlab-Code consists of 299 lines and thus cannot be given here. The analysis of the Gröbner basis solutions shows, that the final univariate polynomial has degree 8, which means that there are up to 8 real solutions to our problem.

3.2.5 Degenerate Configurations

In this section we discuss the degenerate conditions for the proposed algorithms. In previous works [29], [76], [49] the degenerate conditions for the standard 3pt method for essential matrix estimation have been investigated in detail. In these papers multiple degenerate conditions are identified. It is also pointed out that a collinear configuration of 3D points is in general not a degenerate condition for the 3pt method, while it is one for the 5pt method. Degenerate conditions for the standard 3pt algorithm however are collinear points that are parallel to the translation direction and points that are coplanar to the translation vector. We investigated if these scenarios also

	3pt-essential	3pt-hom	2pt	2.5pt
collinear points	no	yes	no	no
collinear points parallel to translation direction	yes	yes	no	no
points coplanar to translation vector	yes	yes	no	no

Table 3.1: Comparison of the degenerate conditions (yes means degenerate) for the standard 3pt method, the proposed 3pt homography method, the 2pt methods and the 2.5pt method.

pose degenerate conditions for our proposed algorithms, the 2pt, 2.5pt and 3pt homography method by conducting experiments with synthetic data. Degenerate cases could be identified by a rank loss of the equation system matrix or for the Gröbner basis case as a rank loss of the action matrix. For the 3pt homography case this revealed that the proposed method shares the degenerate conditions of the standard 3pt method but in addition also has a degenerate condition for the case of collinear points. This is understandable as the 3pt homography method also solves for the plane normal which then has an undefined degree of freedom around the axis of the collinear points. For the 2pt (both 2pt methods share the same properties) and 2.5pt algorithm these special cases however, do not pose degenerate conditions. More information in case of knowledge or partial knowledge of plane parameters allows to avoid degeneracy in the cases critical for the more general 3pt methods. The results of the comparison are summarized in Table 3.1.

3.3 Synthetic Evaluation

To evaluate the algorithms on synthetic data we chose the following setup. The average distance of the scene to the first camera center is set to 1. The scene consists of two planes, one ground plane and one vertical plane which is parallel to the image plane of the first camera. Both planes consist of 200

randomly sampled points. The base-line between two cameras is set to be 0.2, i.e., 20% of the average scene distance, and the focal length is set to 1000 pixels, with a field of view of 45 degrees.
Each algorithm is evaluated under varying image noise and increasing IMU noise. Each of the two setups is evaluated under a forward (along the z-axis) and a sideways (along the x-axis) translation of the second camera. In addition the second camera is rotated around each axis.

To evaluate the robustness of the algorithms we compare the relative translation and rotation separately. The error measure compares the angle difference between the true rotation and the estimated rotation. Since the translation is only known up to scale, we compare the angle between the true- and estimated translation. The errors are computed as follows:

- Angle difference in $\mathbf{R}$:
  $\xi_R = \arccos((\mathrm{Tr}(\mathbf{R}\dot{\mathbf{R}}^\top) - 1)/2)$
- Direction difference in $\mathbf{t}$:
  $\xi_t = \arccos((\mathbf{t}^\top \dot{\mathbf{t}})/(\|\mathbf{t}\|\|\dot{\mathbf{t}}\|))$

Where $\mathbf{R}$, $\mathbf{t}$ denote the ground-truth transformation and $\dot{\mathbf{R}}$, $\dot{\mathbf{t}}$ are the corresponding estimated transformations.

Each data point in the plots represents the 5-quantile[1] (Quintiles) of 1000 measurements.

3.3.1 Relative Pose

Fig. 3.3, Fig. 3.4, and Fig. 3.5 compare the 2-point algorithm to the general 5pt-essential matrix [78], 4pt-homography [44] and 3pt-essential matrix [29] algorithms. Notice, in these experiments the camera poses were computed from points randomly drawn from the ground plane. Since camera poses estimated from coplanar points do not provide a unique solution for the 5pt, 4pt and 3pt-essential matrix algorithm we evaluate each hypothesis with all points coming from both planes. The solution providing the most inliers is chosen to be the correct one. This evaluation is used in all our synthetic experiments.

[1]The k-quantile represents the boundary value of the k^{th} interval when dividing ordered data into k regular intervals. For $k = 2$, the 2-quantile represents the median value.

Similarly Fig. 3.6, Fig. 3.7, and Fig. 3.8 show a comparison of the 2.5pt algorithm with the general 5pt, the 4pt and the 3pt-essential matrix algorithms. Here the camera poses are computed from points randomly sampled from the vertical plane only.
The evaluation shows that knowing the vertical direction and exploiting the planarity of the scene improves motion estimation. The 2pt and 2.5pt algorithms outperform the 5pt and 4pt algorithm, in terms of accuracy. Under perfect IMU measurements the algorithms are robust to image noise and perform significantly better than the 5pt and 4pt algorithm. With increasing IMU noise their performance are still comparable to the 5pt algorithm and superior to the 4pt algorithm.

3.3.2 3pt Homography

Fig. 3.9, Fig. 3.10 and Fig. 3.11 compare the 3pt-homography based algorithm to the general 5pt [78] and the 3pt-essential matrix algorithms [29]. The evaluation shows that the proposed method outperforms the 5pt algorithm, in terms of accuracy. Under perfect IMU measurements the algorithm is robust to image noise and performs significantly better than the 5pt algorithm and equally good as the 3pt-essential matrix algorithm. With increasing IMU noise the performance of the 3pt-essential matrix and 3pt-homography algorithms are still comparable to the 5pt algorithm.

3.3.3 Timings

We evaluate the run-time of all algorithms on an Intel i7-2600K 3.4GHz using Matlab. To provide a fair comparison all algorithms were implemented in Matlab. No mex files were used, except for the reduced row echelon function *rref*, which is required by the 3pt-essential and 3pt-homography algorithms. All timings were averaged over 1000 runs. Table 3.2 summarizes the run-times for each of the six algorithms. The high run time of the 3pt-homography algorithm is due to the complexity of the Grobner basis solution, which has to perform Gauss-Jordan elimination on the 443x451 elimination matrix.

For one RANSAC iteration the timings can vary drastically between algorithms. This is due to the different solution spaces the algorithms provide.

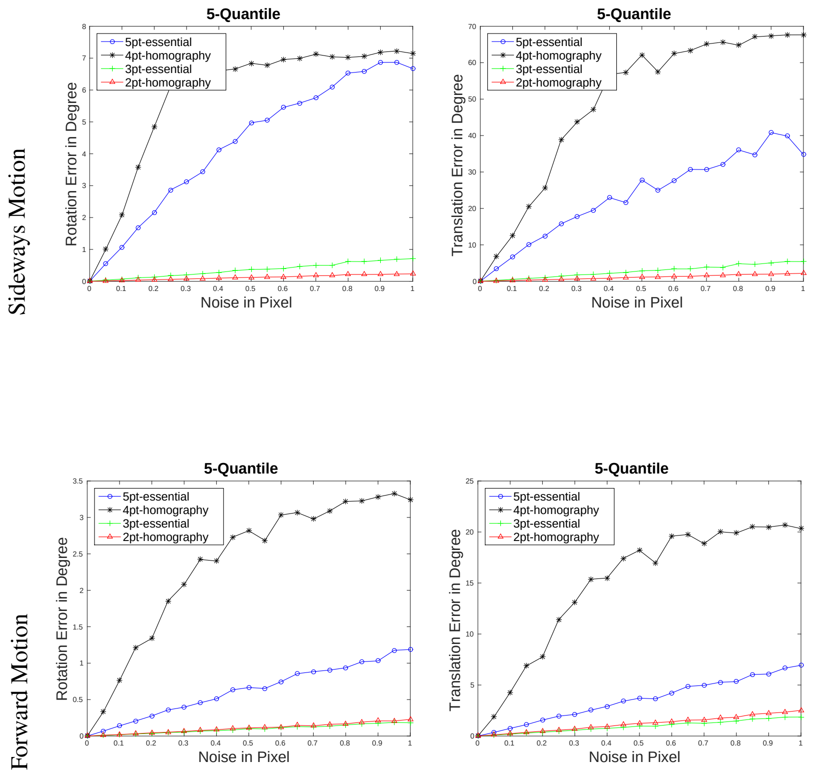


Figure 3.3: Evaluation of the 2 point algorithm under sideways and forward motion with varying image noise.

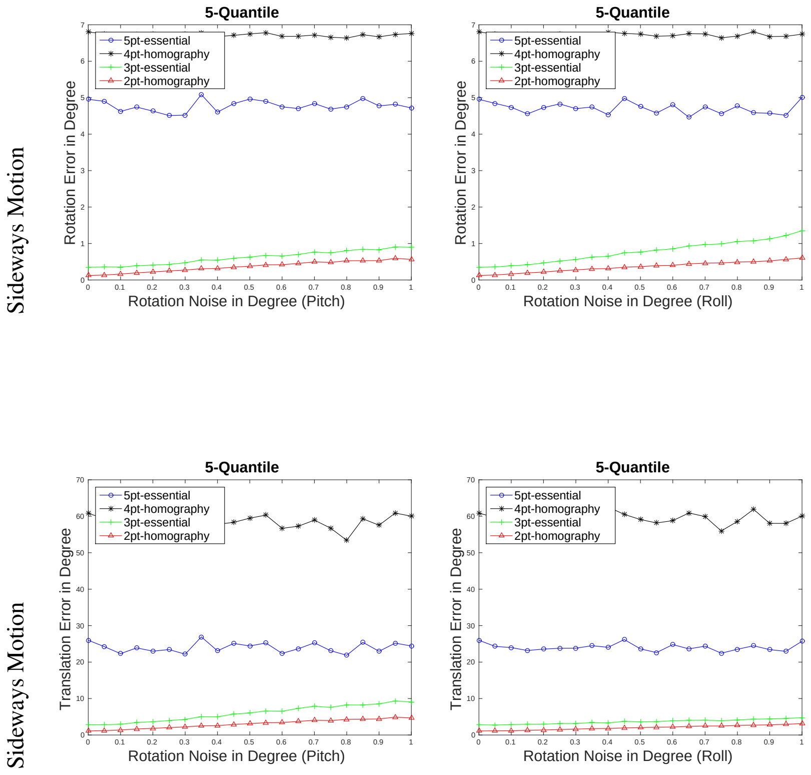


Figure 3.4: Evaluation of the 2pt algorithm under sideways motion using different IMU noise and constant image noise of 0.5 pixel standard deviation. First row rotation error, second row translation error.

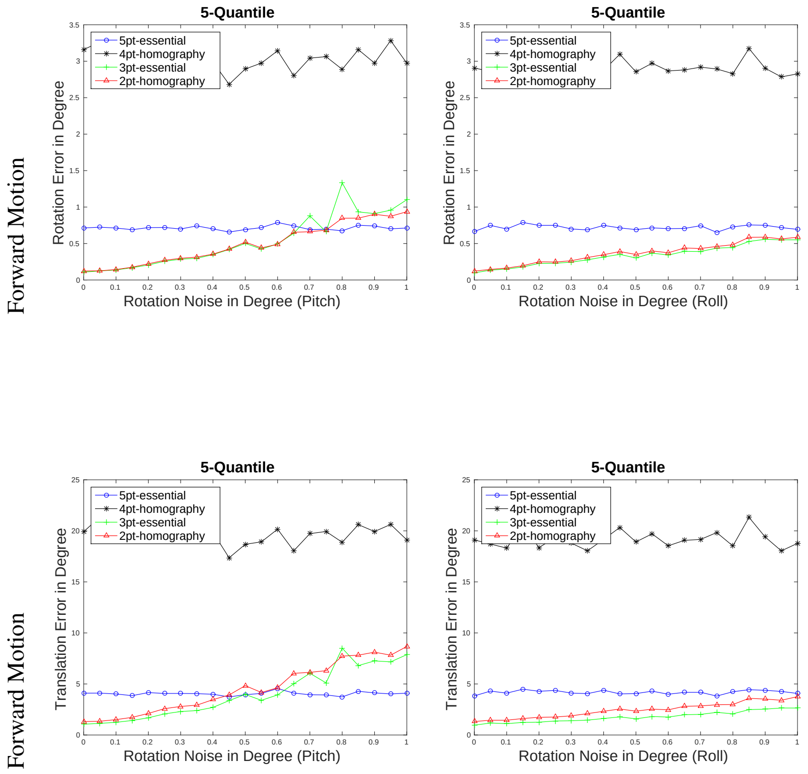


Figure 3.5: Evaluation of the 2pt algorithm under forward motion using different IMU noise and constant image noise of 0.5 pixel standard deviation. First row rotation error, second row translation error.

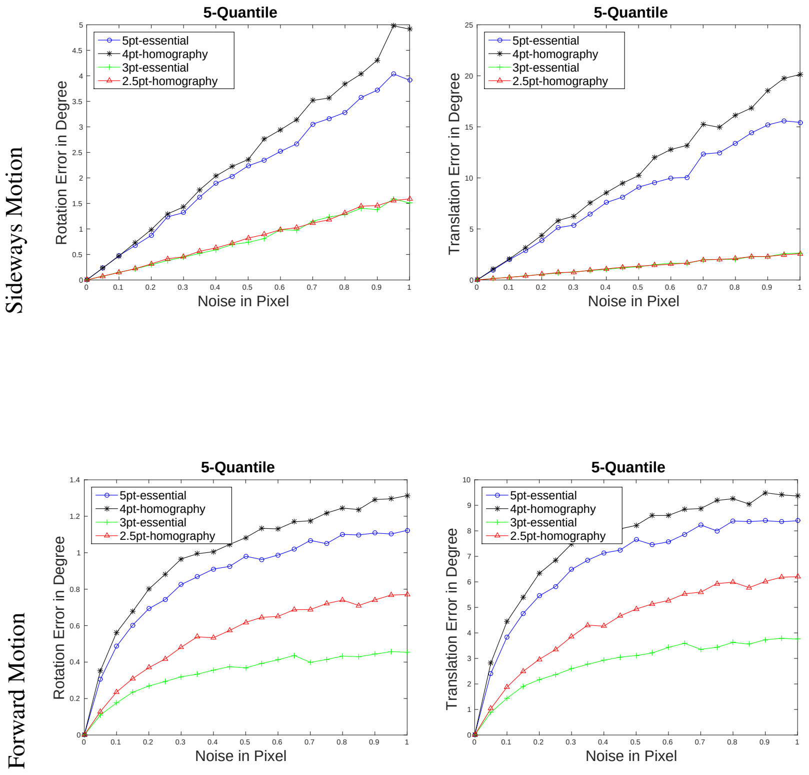


Figure 3.6: Evaluation of the 2.5pt algorithm under forward and sideways motion with varying image noise.

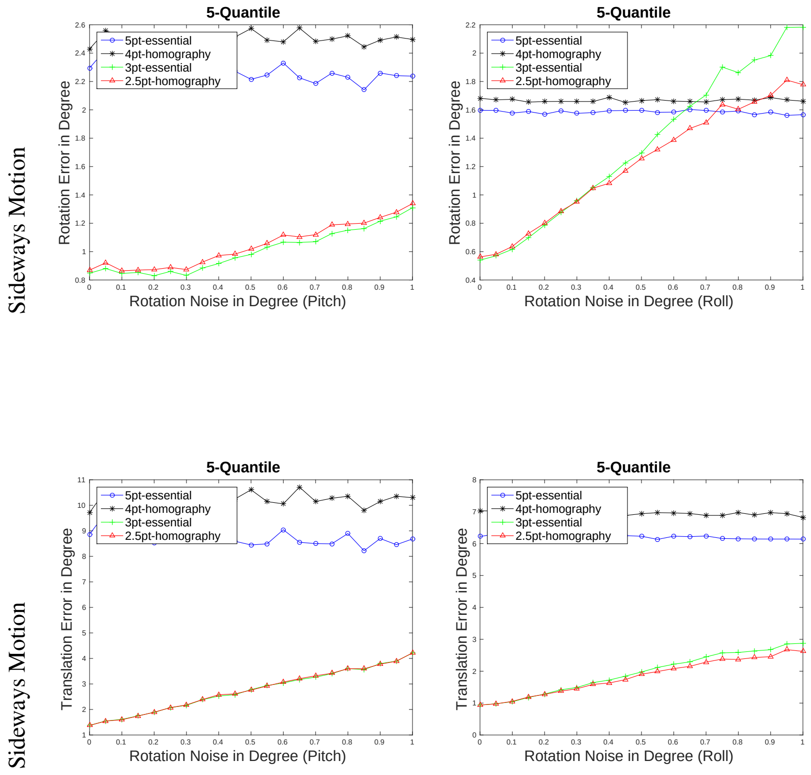


Figure 3.7: Evaluation of the 2.5pt algorithm under sideways motion, with different IMU noise and constant image noise of 0.5 pixel standard deviation. First row rotation error, second row translation error.

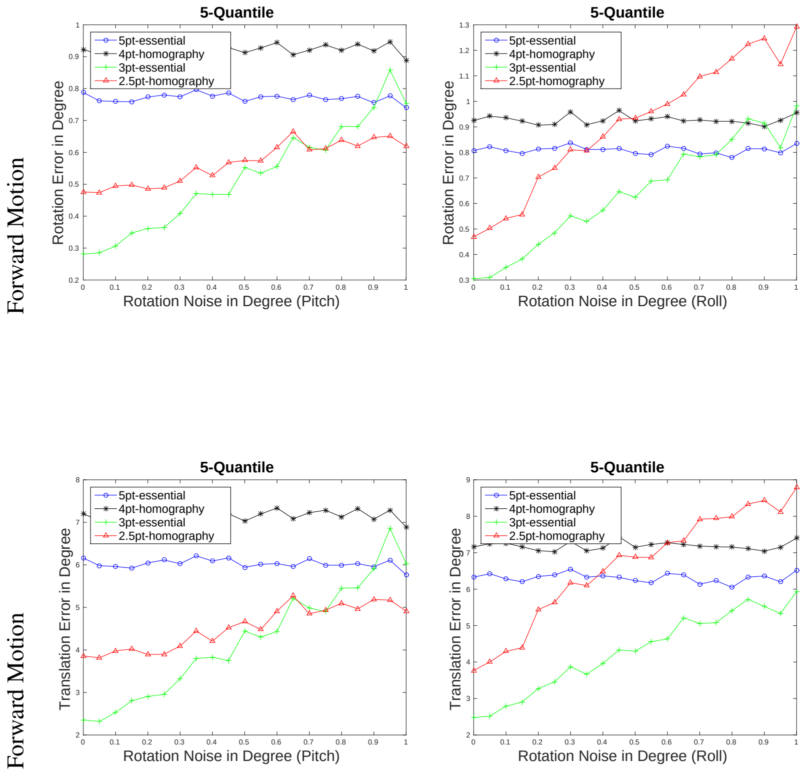


Figure 3.8: Evaluation of the 2.5pt algorithm under forward motion, with different IMU noise and constant image noise of 0.5 pixel standard deviation. First row rotation error, second row translation error.

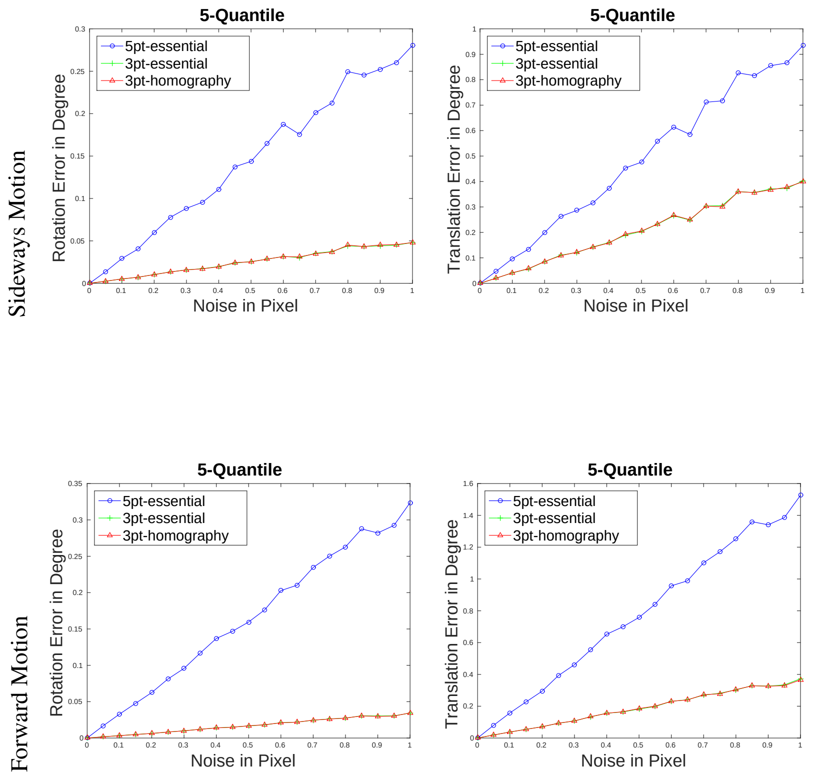


Figure 3.9: Evaluation of the 3pt-homography algorithm under sideways and forward motion with varying image noise.

To have the same error measure for all algorithms, we choose to use the re-projection error to select the correct camera poses among a set of possible poses. For instance the 2pt algorithm provides one unique camera pose, while the 5pt algorithm can provide up to 10 different essential matrices. In addition for each essential matrix 4 possible camera poses exist and need to be verified to find the correct pose, which can result in a total of 40 possible camera poses. While the homography formulations directly provide sets of camera poses. Even though the hypothesis estimation of the 3pt-homography algorithm has a larger constant time complexity, compared to its essential matrix counter part, one RANSAC iteration is cheaper, since fewer potential poses need to be evaluated. The table clearly shows that the computation time is dominated by the hypothesis selection (re-projection error computation) and not by the solver. In all experiments we used a set of 200 point correspondences.

Methods	Hypothesis Estimation(ms)	RANSAC 1 Iteration(ms)	Avg. # Solutions
2pt	0.09	8.31	2
2.5pt	0.22	33.45	8
3pt-homography	27.28	55.17	6.85
3pt-essential	0.49	25.02	6.18
4pt-homography	0.18	8.65	2
5pt-essential	0.42	64.33	16.02

Table 3.2: Run-time comparison of different pose estimation algorithms. The second column provides timings for estimating the hypothesis. The third column provides timings for one RANSAC iterations, which includes the selection of the right solution from a set of hypothesis. The last column shows the average number of real solutions (camera poses) provided by the respective algorithm. See text for more details.

3.4 Real Data Experiments

In the following section we evaluate the proposed algorithms on both an indoor and outdoor environment.

3.4.1 Error Measure

In order to compare the estimated camera poses to the ground-truth, we used the relative pose error (RPE) measure as proposed by Sturm [101]. The RPE compares the local accuracy of the trajectory over a fixed time interval Δ, that corresponds to the drift of the trajectory. The RPE at time step i can be defined as:

$$\mathbf{E}_i = (\mathbf{Q}_i^{-1}\mathbf{Q}_{i+\Delta})^{-1}(\mathbf{P}_i^{-1}\mathbf{P}_{i+\Delta}), \tag{3.42}$$

where $\mathbf{Q}_i, \mathbf{P}_i \in SE(3)$ represent the ground truth and estimated poses respectively. $\mathbf{E}_i$ then represents the relative error. For a sequence of n camera poses, $m = n - \Delta$ individual relative pose errors are then estimated. From these errors, we propose to compute the root mean squared error (RMSE) over all time indices of the translational component as

$$RMSE(\mathbf{E}_{1:n}, \Delta) = \sqrt{\left(\frac{1}{m}\sum_{i=1}^{m} \|trans(E_i)\|^2\right)}, \tag{3.43}$$

where $trans(\mathbf{E}_i)$ refers to the translational components of the relative pose error $\mathbf{E}_i$.

3.4.2 Vicon Dataset

In order to have a practical evaluation of the 2pt, 2.5pt and 3pt algorithms, several real datasets have been collected with reliable ground-truth, see Fig. 3.12. The ground-truth data has been obtained by conducting the experiments in a room equipped with a Vicon motion capture system made of 22 cameras. We used the Vicon data as inertial measures and scale factor in the different experiments. The sequences have been acquired with a perspective camera mounted either on teleoperated Segway mobile robot (Fig. 3.12) or with a handheld system in order to have planar and 3D trajectories. In both cases the cameras are synchronized with the Vicon system. The image resolution used is 1624×1234 pixels. The length of these trajectories is between 20 and 50 meters and the number of images is between 150 and 350 per sequence. Robot motion speed is about 1m/s. Two different sets have been acquired, one set showing the ground plane dominantly and another set showing the walls dominantly.

We perform a comparison of the 2pt, 2.5pt and 3pt-homography with the 5pt algorithm in order to show the efficiency of the proposed methods. First, we use [111] to extract and match SIFT [68] features. The same matched feature point sets are used for the different algorithms and form the input to RANSAC [26] in order to select the inliers. For RANSAC we use a fixed number of 100 iterations, in all our experiments.

Figure 3.13 - Fig. 3.15 shows the evaluation of the 2pt ground plane algorithm. The trajectories obtained with 2pt (red curve) and 5pt (black curve) are compared with the ground-truth (blue curve) from Vicon. In all these experiments, even if both approaches propose trajectories globally with a similar shape than the ground-truth, we can note that the 2pt algorithm provides better results than the 5pt method. In the case of planar trajectories, it is worth noting that the 2pt algorithm has a very low drift in the vertical axis while the 5pt accumulates significant error. Over the six sequences, the mean angular error in translation is equal to 0.1883 radians for the 2pt and 0.3380 for the 5pt.

The root mean squared error as defined in equation 3.43 is given for all 6 sequences in table 3.3. The 2pt clearly outperforms the 5pt algorithm, providing a $1.69 \times -6.54\times$ lower error compared to the 5pt algorithm.

Figure 3.16 compares the different trajectories obtained from the 2.5pt (red curve), the 3pt-homography algorithm (green curve), the 3pt-essential matrix algorithm (magenta curve) and the 5pt (black curve), to the Vicon ground-truth (blue curve).

The 2.5pt algorithm shows similar performance compared to the standard 5pt algorithm and both 3pt algorithms, however having the advantage of a much simpler derivation. This experiments also demonstrated that the assumptions taken for the 2pt algorithm (flat ground plane) and for the 2.5pt algorithm (vertical walls) are met in practical situations and can be used in

	2pt (mm)	5pt (mm)
Ground Sequence I	8.94	48.55
Ground Sequence II	9.28	56.66
Ground Sequence III	18.46	65.39
Ground Sequence IV	14.25	93.30
Ground Sequence V	25.61	34.46
Ground Sequence VI	39.75	67.45

Table 3.3: Root Mean Squared Error overview.

real applications.

The computation time was estimated for each pair of images during a complete sequence. Only the part dedicated to the robust estimation of the visual odometry based on 2pt and 5pt algorithms was evaluated. For both algorithms, the same set of matched points was proposed as entry to the estimation. The mean computation times over the complete sequence were respectively equal to 14.75 seconds for the 5pt algorithm and 0.03 for the 2pt algorithm.

3.4.3 2pt Algorithm in a SFM Pipeline

In this final experiment we demonstrate the usage of the 2pt algorithm within an incremental SFM pipeline. The 2pt algorithm is used to replace the 5pt algorithm within the SFM pipeline. For this experiment the MAVMAP [99] SFM pipeline has been adapted to compare the 2pt algorithm to the 5pt algorithm. Two-view pose estimation is used when processing each new frame. To compute the relative pose between two consecutive frames we estimate the essential matrix in case of the 5pt algorithm and the homgography for the the 2pt algorithm. Afterwards full bundle adjustment is performed to compute precise camera poses and 3D points. The main goal of this experiment is to show that the 2pt can in practice replace standard algorithms (like the 5pt) for gaining a speed up but by maintaining the accuracy of the system.

For this experiment a UAV data set of a parking lot (denotated ParkingLot data set) is used. The image resolution for this data set is 24MP. The UAV was equipped with GPS and the GPS trajectory is utilized as ground truth for comparison. Figure 3.17 shows the results for this experiment. Figure 3.17(a) shows the output of the SFM system, resulting 3D point cloud (densified with SURE [69]), camera positions (red) and GPS positions (green). Figure 3.17(b) shows RPE plots for an experiment using the 5pt (black) algorithm and the 2pt (red) algorithm. Both algorithms lead to almost identical results. The value of the remaining RPE error is mainly due to the uncertainty of the GPS measurements and expected in this form. The resulting re-projection error after bundle adjustment is 0.249px for the 2pt case and 0.246px for the 5pt case, almost identical. To be clear, the reason for the identical re-projection error comes from bundle adjustment. This experiment demonstrates that the proposed 2pt algorithm can successfully replace the standard 5pt in a SFM system seamlessly but with the advantage of a gained speed-up.

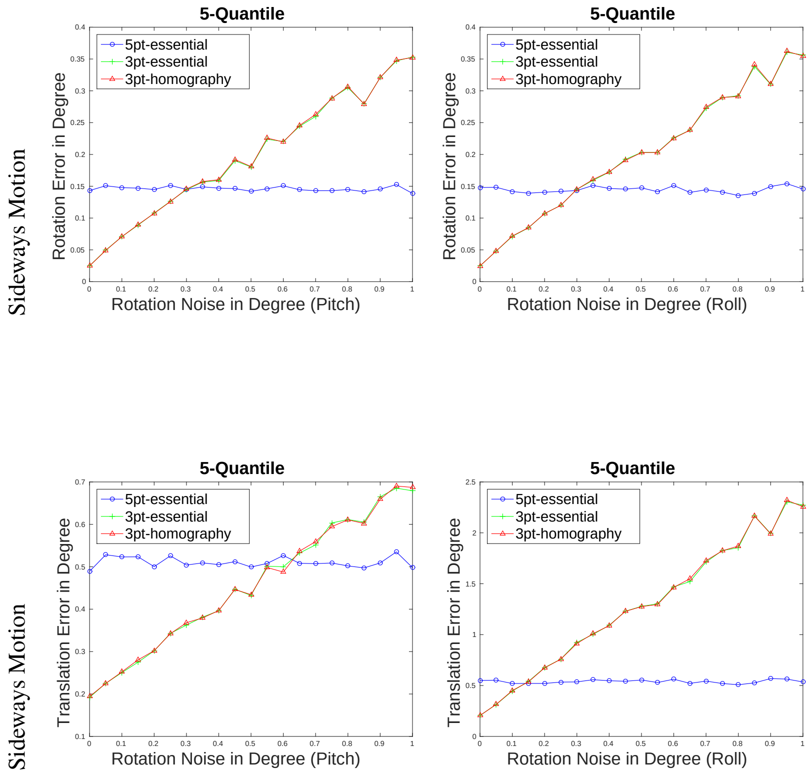


Figure 3.10: Evaluation of the 3pt-homography algorithm under sideways motion, with different IMU noise and constant image noise of 0.5 pixel standard deviation.First row rotation error, second row translation error.

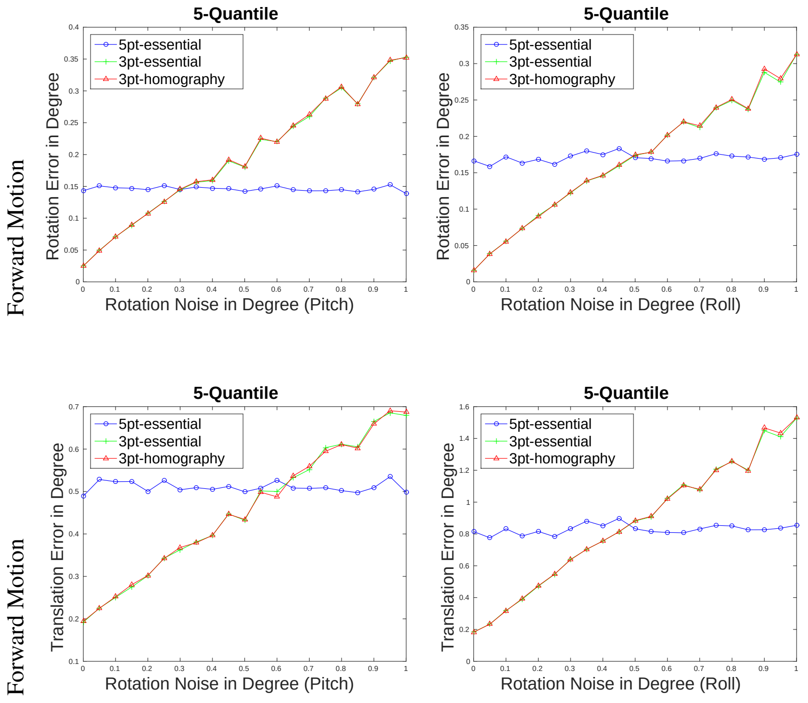


Figure 3.11: Evaluation of the 3pt-homography algorithm under forward motion, with different IMU noise and constant image noise of 0.5 pixel standard deviation. First row rotation error, second row translation error.

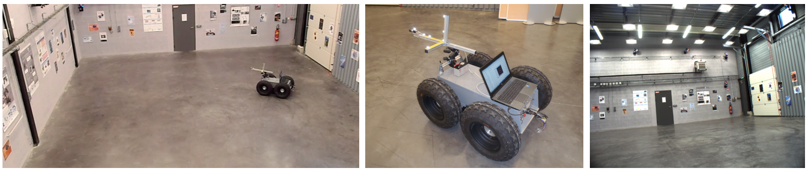


Figure 3.12: Left, Vicon arena used to record the ground-truth dataset. Center, teleoperated Segway mobile robot capturing data. Right, sample image captured by the robot.

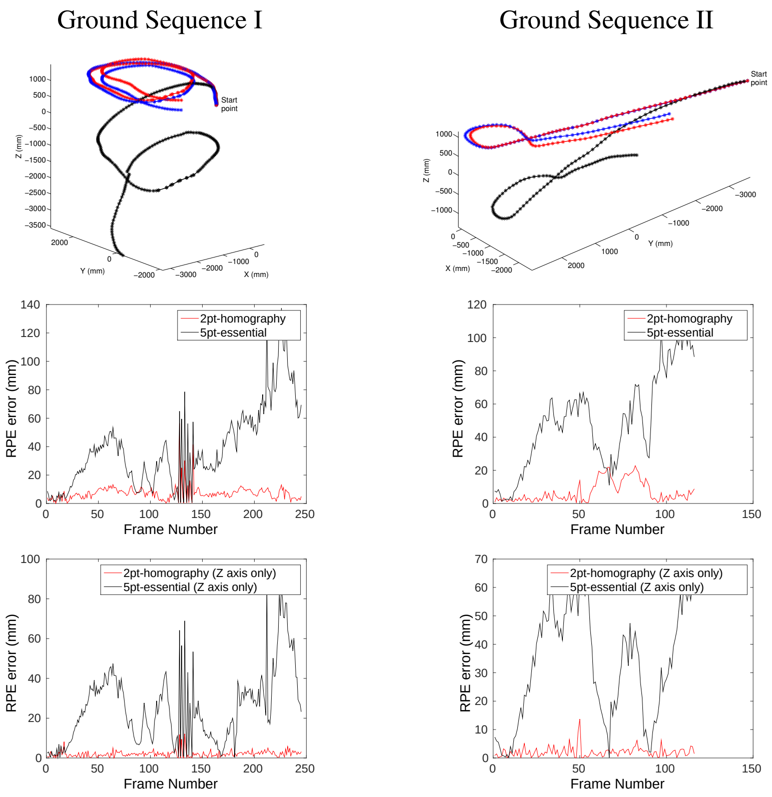


Figure 3.13: Evaluation of the 2pt ground plane algorithm: Top row visual odometry estimated using the 2pt (red) and the 5pt (black) algorithm. The Vicon ground-truth is given in blue. Middle row, the Relative Pose Error (RPE) in *mm* for each individual frame. Bottom row, shows the RPE error for the vertical axis (z-axis).

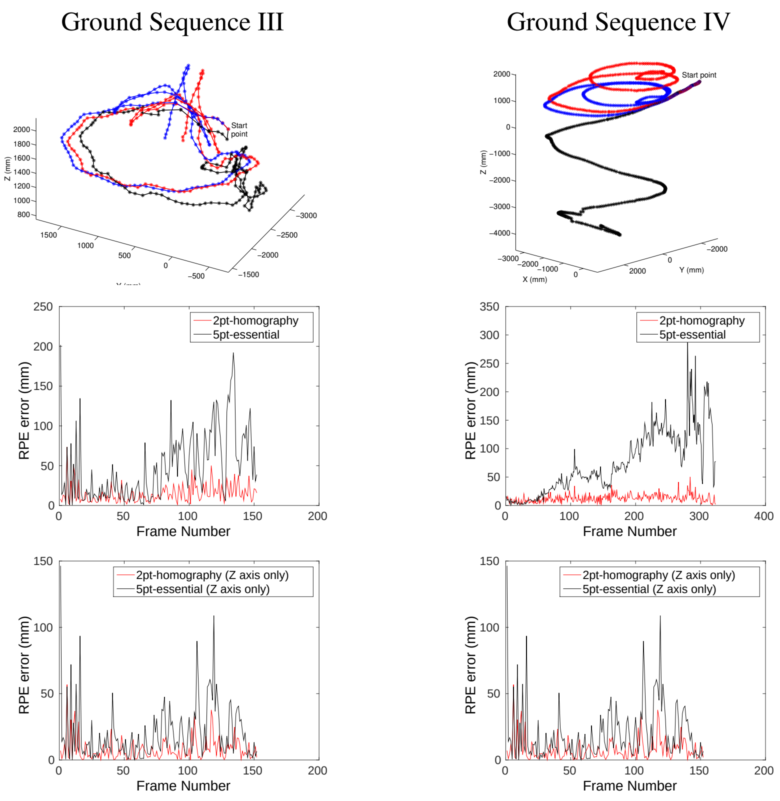


Figure 3.14: Evaluation of the 2pt ground plane algorithm: Top row visual odometry estimated using the 2pt (red) and the 5pt (black) algorithm. The Vicon ground-truth is given in blue. Middle row, the Relative Pose Error (RPE) in *mm* for each individual frame. Bottom row, shows the RPE error for the vertical axis (z-axis).

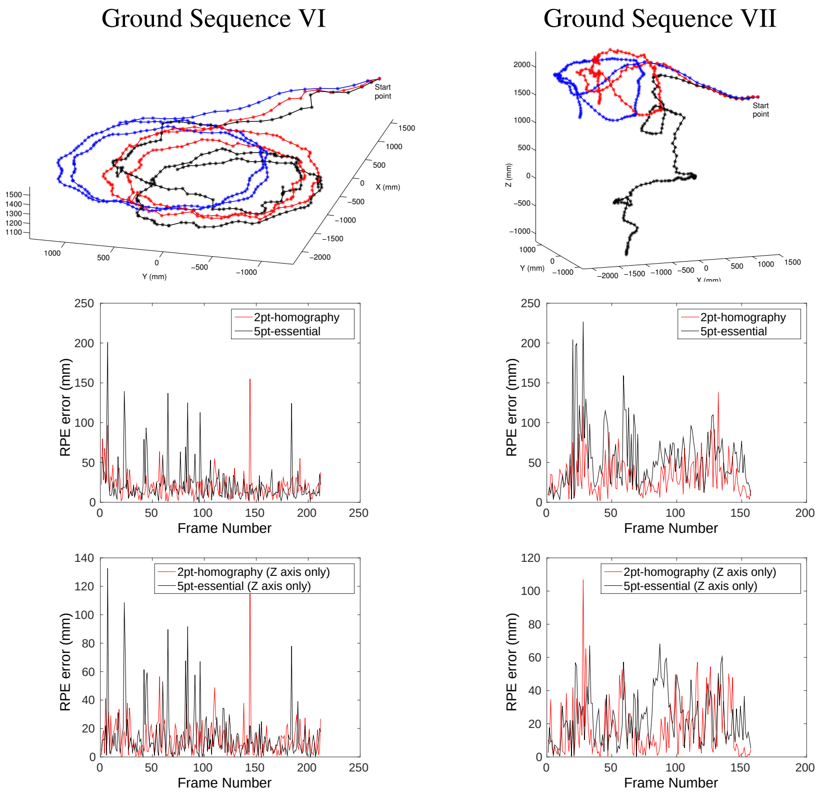


Figure 3.15: Evaluation of the 2pt ground plane algorithm: Top row visual odometry estimated using the 2pt (red) and the 5pt (black) algorithm. The Vicon ground-truth is given in blue. Middle row, the Relative Pose Error (RPE) in *mm* for each individual frame. Bottom row, shows the RPE error for the vertical axis (z-axis).

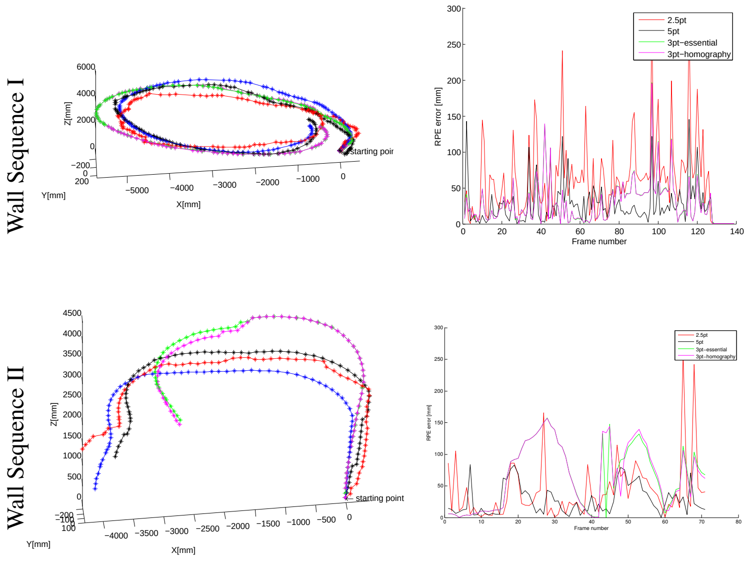
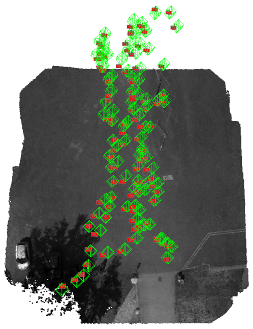
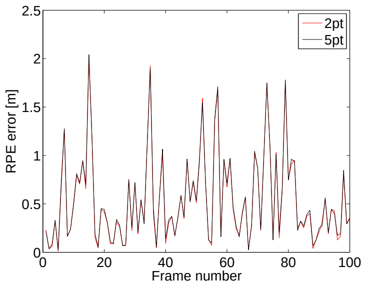


Figure 3.16: Evaluation of the 2.5pt vertical wall algorithm: Trajectories estimated with 2.5pt (red), with 3pt-homography (green curve), with 3pt-essential matrix (magenta) and 5pt (black curve) algorithms compared with the Vicon ground-truth (blue curve).



(a)



(b)

Figure 3.17: Results of an incremental SFM pipeline using the 2pt algorithm. (a) Resulting 3D point cloud, camera positions (red) and GPS positions (green). (b) RPE error plot when using 5pt or 2pt within the SFM pipeline. The initial solution of the 5pt and 2pt are similar enough for bundle adjustment to converge to almost the same final solution.

4 Rolling Shutter Pose Estimation

Contents

Absolute pose estimation in Computer Vision refers to the problem of finding the camera pose in the world frame given a set of 3D scene points expressed in the world frame and corresponding set of 2D image points expressed in the camera coordinate frame. A minimum of three 2D-3D correspondences are needed to solve for the absolute pose in the case of a global shutter camera. This is commonly referred to as the Perspective-3-Point or P3P problem [42]. A generalization of the P3P problem to n-point correspondences is known as the PnP problem. The solution to the absolute pose estimation problem has great importance in performing robotics visual Simultaneous Localization and Mapping (visual SLAM), localization with respect to a given map, and Structure-from-Motion (SfM).

Over the years, a huge literature of solutions to the absolute pose estimation problem [42, 56, 57, 87] have been developed by many researchers for the global shutter camera. The solutions to the absolute pose estimation problem for global shutter cameras, however, do not work equally well for a moving rolling shutter camera. This is because the existing solutions are modeled for global shutter cameras that take snapshots of a scene by exposing the entire photo-sensor in a single instance of time. These solutions do not account for

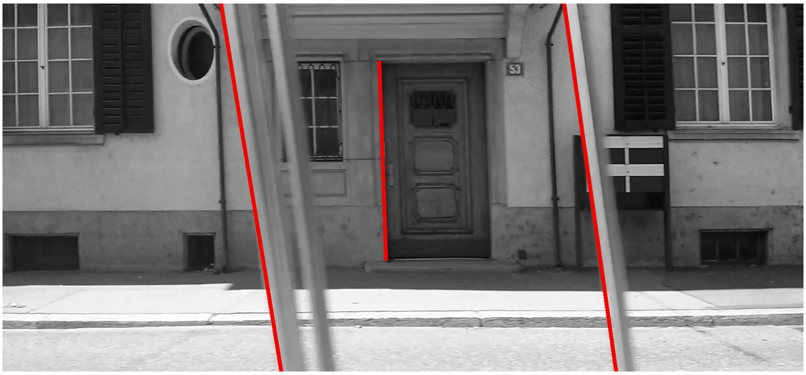


Figure 4.1: Artifacts on an image taken with a rolling shutter camera moving towards the left of the image. The poles and and door frame (marked in red), which are supposed to be upright, appear slanted to the left due to the camera motion and the sequential exposure of each scanline from top to bottom of the image. Objects further away from the camera are less affected by the rolling shutter.

the image artifacts caused by a moving rolling shutter camera that simultaneously exposes scanlines of its photo-sensor either horizontally or vertically over a rapid instance of time ($\sim$ 72ms in our real data experiments). Figure 4.1 shows an example of an image taken by a moving rolling shutter camera. The camera moves towards the left of the image and the scanlines progress from top to bottom of the image. As a result, scene objects such as the fence and building facade edge (marked in red) appear to be slanted to the left of the image.

While there is an inherent difficulty in doing absolute pose estimation with rolling shutter cameras, there is already a widespread usage of the rolling shutter cameras due to the low cost in manufacturing and robustness of the CMOS photo sensors, and the massive incorporation of the cameras into mobile devices such as mobile phones and tablets. It is therefore useful to provide an algorithm that corrects for the moving rolling shutter camera artifacts while doing absolute pose estimation.

In this chapter, we propose a minimal solution to the rolling shutter camera pose estimation problem. In particular, we introduce an additional linear velocity in the camera projection matrix to model the motion of the rolling shutter camera. We noted that this assumption holds in practice because the scanline speed is always much faster than the velocity of the rolling shutter camera mounted on a hand-held mobile phone, tablet or a moving car. We show that a minimum of 5-point 2D-3D correspondences are needed to solve for the pose and linear velocity using the Gröbner basis [22] and gives up to eight real solutions. We also show that our formulation can be extended to use more than 5-point correspondences. We use RANSAC [26] for robust estimation to get all the inlier point correspondences. We also identify the correct solution from the eight possible solutions within RANSAC. Finally, we relax the linear velocity assumption and do a non-linear refinement on the full motion, i.e. linear and angular velocities, and pose of the rolling shutter camera with all the inliers. We verify the feasibility and accuracy of our algorithm with both simulated and real-world datasets.

4.1 Related Work

Most of the existing works on rolling shutter cameras largely revolve around calibration, correction for rolling shutter distortion on the images, using rolling shutter cameras for stereo setups, and iterative methods for pose estimation. In contrast, we propose a minimal solution to estimate the rolling shutter camera pose and velocity in this work. Our minimal solution requires only 5-point correspondences and this makes it very suitable to be used within RANSAC for robust estimation to find all the inlier correspondences.

One of the early publications on rolling shutter camera is from Liang *et al.* [62]. They gave detailed discussions on the rolling shutter effect and low level CMOS sensor that usually has an electronic rolling shutter. In this work, the authors proposed to compensate for the rolling shutter effect using optical flow. In [37], Geyer *et al.* proposed a method to calibrate the rolling shutter timings using additional hardware and studied the different rolling shutter effects under special fronto-parallel motion. More recently, Oth *et al.*

proposed in [83] to calibrate the shutter timings using a video sequence of a known calibration pattern. A continuous-time trajectory model is combined with a rolling shutter model to estimate the shutter timings.

In [10, 16], the authors proposed 2D approaches for rolling shutter image stabilization and rolling shutter distortion correction using optical flow. Similarly, [39] used optical flow and a mixture of homographies to correct for the rolling shutter effect. In [8, 41], Hanning *et al.* and Karpenk *et al.* proposed rolling shutter distortion correction base on gyroscope measurements. Their assumption is that on hand-held devices the main motion during exposure is due to a rotation and can be compensated with a homography.

While the above approaches are 2D in nature, Forssen and Ringaby [27, 88] proposed a Structure-from-motion approach to compensate for the rolling shutter distortion that is mainly induced by rotational motions. In [45], Hedborg *et al.* proposed a full rolling shutter bundle adjustment on a continuous video stream by enforcing a continuous pose parametrization between consecutive frames. Klein *et .al* proposed in [50] to first estimate a constant velocity between consecutive frames and uses this motion model to undo the RS distortion on the extracted keypoints. The corrected keypoints are then used in a standard bundle adjustment [106] for global shutter camera.

In [6,92], the authors have proposed stereo algorithms that take into account the rolling shutter model and produces geometrically consistent 3D reconstructions. In [73], Meilland *et al.* proposed a dense 3D model registration which accounts for rolling shutter distortion and motion blur on RGBD data.

Probably closest to our work is the work by Ait-Aider *et al.* [4], where they estimate the pose and velocity of a moving object from a single rolling shutter image. They use a spiral motion parametrization of the camera pose and solve for the pose and velocities as a non-linear least squares problem. Their formulation requires a good initialization which is obtained from a global shutter pose algorithm. The 3D-2D correspondences are provided manually. In [5], the authors extended the initialization process with a homography based formulation, which takes into account the temporal pose parametrization. It is however limited to only planar objects. To overcome the initialization burden, Magerand *et. al* [70] solved the pose and velocities using constrained global optimization by parameterizing the camera motion with degree 2 polynomials. The final objective function they need to solve for consists of a 6 degree polynomial with twelve unknowns.

4.2 Rolling Shutter Absolute Pose

4.2.1 Camera Motion Model

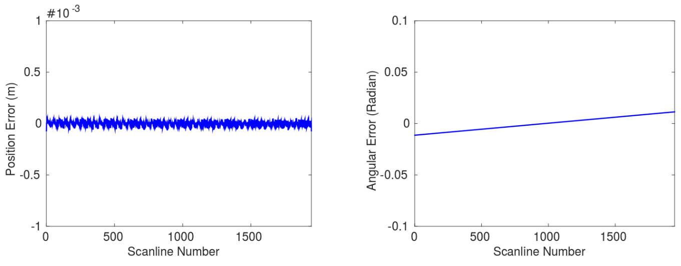
Since a rolling shutter camera typically has a rapid scanning time ($\sim$ 72ms per image), it is reasonable to make the assumption that the camera undergoes constant linear and angular velocities during an image acquisition. We further assume that each scanline takes exactly the same time, and the relative camera translation $\mathbf{t}_n$ and rotation $\mathbf{R}_n$ at the n^{th} scanline with respect to the first scanline can be linearly interpolated as

$$\mathbf{t}_n = \mathbf{v}n\tau \tag{4.1a}$$

$$\mathbf{R}_n = \mathbf{exp}(\boldsymbol{\Omega}n\tau), \tag{4.1b}$$

where $\mathbf{v} = [v_x, v_y, v_z]^\top$ and $\boldsymbol{\Omega} = [\omega_x, \omega_y, \omega_z]^\top$ denotes the constant linear and angular camera velocities, and τ is the time taken to complete each scanline. The function $\mathbf{exp}(.) : so(3) \rightarrow SO(3)$ denotes the exponential map that transforms the angle-axis rotation representation to a corresponding rotation matrix.

As mentioned in Section 4 that in practice the scanline speed is always much faster than the velocity of the camera, the camera motion can be approximated with only the linear velocity. As such, we consider only the linear velocity in our derivation of the minimal solution, i.e. $\mathbf{R}_n = \mathbf{exp}(0) = \mathbf{I}_{3\times 3}$. We justify the validity of this assumption with the results from a real data experiment. We look at an image sequence where a car with a rolling shutter camera mounted on it takes a 90° turn, while driving at 10km/h. During the scan time of the CMOS sensor (72ms in our case), the car moved 0.2m and the absolute camera orientation changed by 0.02rad. Figure 4.2 compares the ground truth GPS/INS poses to the interpolated poses assuming zero angular velocity. The maximum absolute angular error obtained is only 0.01rad. We will further discuss the valid range of this assumption in Section 4.3.1.



(a) (b)

Figure 4.2: The plots show the position and rotation error while the car moves through a $90°$ turn. During image scan time of 72ms the car moved 0.20m. (a) The maximum position error is 1.8718×10^{-4}m. (b) Maximum rotation error assuming zero angular velocity is 0.0114 rad.

4.2.2 Minimal 5-Point Algorithm

Making the assumption of a constant linear velocity, we can express a pixel on the n^{th} scanline as

$$\mathbf{x}_n = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t} - \mathbf{t}_n\end{bmatrix}\mathbf{X}, \tag{4.2}$$

where $\mathbf{K}$ is the camera intrinsic. $\mathbf{R}$ and $\mathbf{t}$ are the camera pose in the world frame, which is also the camera pose for the first scanline. $\mathbf{t}_n$ is the camera pose for the n^{th} scanline as given in Equation 4.1. $\mathbf{x}_n \leftrightarrow \mathbf{X}$ is the 2D-3D point correspondence. Formally, Equation 4.2 is the camera projection equation that accounts for the rolling shutter effect. The unknowns are $\mathbf{R}$, $\mathbf{t}$ and the linear velocity $\mathbf{v}$ in $\mathbf{t}_n$, where there are altogether 9 degree-of-freedom (3 degree-of-freedom each for $\mathbf{R}$, $\mathbf{t}$ and $\mathbf{v}$). Since each point correspondence gives two independent equations, a minimum of 5-point correspondences are needed to solve for all the unknowns in Equation 4.2. Taking the cross product of $\mathbf{x}_n$ with Equation 4.2, we get

$$\mathbf{x}_n \times (\mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}-\mathbf{t}_n\end{bmatrix}\mathbf{X}) = 0. \tag{4.3}$$

With 5-point correspondences, Equation 4.3 can be rearranged into the form

$$\mathbf{Ay} = \mathbf{0}, \tag{4.4}$$

where **A** is a matrix made up of the known values from the camera intrinsic **K**, point correspondences $\mathbf{x}_n \leftrightarrow \mathbf{X}$, scanline number n and time τ. Here we choose randomly 9 out of the 10 equations in the minimal 5-point correspondence case (since any 9 out of the 10 equations are always independent) to form the 9×15 matrix **A**.

$$\mathbf{y} = \begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & t_x & t_y & t_z & v_x & v_y & v_z\end{bmatrix}^\top \tag{4.5}$$

is a 15×1 vector, where $\mathbf{r}_i$ is the i^{th} row of the rotation matrix **R**, $[t_x\ t_y\ t_z]$ are the components from the translation vector **t** and $[v_x\ v_y\ v_z]$ are from the linear velocity **v** of the camera. Solving for the right nullspace of **Ay = 0** using the Singular Value Decomposition (SVD) gives 6 basis vectors denoted by $\mathbf{b}_1...\mathbf{b}_6$, where the linear combination forms the solution

$$\mathbf{y} = \beta_1\mathbf{b}_1 + \beta_2\mathbf{b}_2 + \beta_3\mathbf{b}_3 + \beta_4\mathbf{b}_4 + \beta_5\mathbf{b}_5 + \beta_6\mathbf{b}_6. \tag{4.6}$$

$\beta_1...\beta_6$ are any scalar values. Assigning random values to $\beta_1...\beta_6$ however do not guarantee the orthogonality of **R**. We fix $\beta_6 = 1$ and the remaining five scalar values can be found by enforcing the orthogonal constraint on the elements from the rotation matrix **R**. Following [112], enforcing the orthogonality on **R** gives us 10 constraints:

$$||\mathbf{r}_1||^2 - ||\mathbf{r}_2||^2 = 0, \tag{4.7a}$$

$$||\mathbf{r}_1||^2 - ||\mathbf{r}_3||^2 = 0, \tag{4.7b}$$

$$||\mathbf{c}_1||^2 - ||\mathbf{c}_2||^2 = 0, \tag{4.7c}$$

$$||\mathbf{c}_1||^2 - ||\mathbf{c}_3||^2 = 0, \tag{4.7d}$$

$$\mathbf{r}_1{}^\top\mathbf{r}_2 = 0,\ \mathbf{r}_1{}^\top\mathbf{r}_3 = 0,\ \mathbf{r}_2{}^\top\mathbf{r}_3 = 0, \tag{4.7e}$$

$$\mathbf{c}_1{}^\top\mathbf{c}_2 = 0,\ \mathbf{c}_1{}^\top\mathbf{c}_3 = 0,\ \mathbf{c}_2{}^\top\mathbf{c}_3 = 0, \tag{4.7f}$$

which we can use to solve for the scalar values $\beta_1 ... \beta_5$ that formed the solution. $\mathbf{r}_i$ denotes the i^{th} row and $\mathbf{c}_i$ the i^{th} column of $\mathbf{R}$. Putting the elements from $\mathbf{R}$ in Equation 4.6 into the 10 constraints, we get a system of 10 polynomial equations with $\beta_1 ... \beta_5$ as the unknowns. We use the automatic generator of Gröbner solvers provided by Kukelova $et\ al.$ [53] to generate a solver for the system of polynomial equations that gives up to eight real solutions for $\mathbf{y}$. We divide each of the solution by its respective $||\mathbf{r}_1||$ to make $\mathbf{R}$ an orthonormal matrix. Note that the orthonormal constraint is not enforced earlier in Equation 4.7 to keep the system of polynomials less complicated. In addition, we ensure that the solution follows a right-hand coordinate system by negating the solution if $\det(\mathbf{R}) = -1$.

It should be noted that our formulation also works for m-point correspondences where $m \geq 5$, i.e. ≥ 10 independent equations are used to form Equation 4.4. In this case, we get an over-determinate system. The 6 basis vector in Equation 4.6 can be obtained from the 6 singular vectors that correspond to the 6 smallest singular values of $\mathbf{A}$.

4.2.3 Robust Estimation

We use the minimal 5-point algorithm within RANSAC [26] to robustly select all the inlier 2D-3D correspondences. We also determine the correct solution from the 8 solutions within each RANSAC loop as the one that gives the most inlier count.

4.2.4 Non-linear Refinement

The 5-point minimal solver with RANSAC provides an initial solution and finds all the inlier 2D-3D correspondences, which is then used for further refinement using a non-linear solver [2]. Here, we relaxed the linear velocity assumption and do a refinement on the full camera motion, i.e. linear $\mathbf{v}$ and angular $\boldsymbol{\Omega}$ velocities, and pose ($\mathbf{R}$, $\mathbf{t}$). Formally, we seek to minimize the total reprojection errors over $\mathbf{v}$, $\boldsymbol{\Omega}$, $\mathbf{R}$ and $\mathbf{t}$. The objective function is given by

$$\underset{\mathbf{v},\boldsymbol{\Omega},\mathbf{R},\mathbf{t}}{\operatorname{argmin}} \sum_i ||\mathbf{x}_{n,i} - \pi(\mathbf{P}_{n,i}, \mathbf{X}_i)||^2, \tag{4.8}$$

where $\mathbf{x}_{n,i}$ is the i^{th} 2D image point on the n^{th} scanline, $\mathbf{X}_i$ is the corresponding 3D point, $\pi(.) = \mathbf{P}_{n,i}\mathbf{X}_i$ is the reprojection function. $\mathbf{P}_{n,i}$ is the rolling shutter camera projection matrix for the n^{th} scanline given by

$$\mathbf{P}_{n,i} = \mathbf{K}\mathbf{R}_n \begin{bmatrix}\mathbf{R} & \mathbf{t} - \mathbf{t}_n\end{bmatrix}, \tag{4.9}$$

where ($\mathbf{t}_n$, $\mathbf{R}_n$) is the pose of the rolling shutter camera when the n^{th} scanline was taken as defined in Equation 4.1.

4.3 Algorithm Evaluation

4.3.1 Synthetic Data

We first evaluate our proposed algorithm on several synthetic configurations. Specifically, we do comparisons for the following three methods:

1. **GS + refinement**: Global shutter P3P [56] with non-linear rolling shutter aware refinement of pose and velocities as described in Section 4.2.4. Note that in a perspective configuration the generalized P3P algorithm [56] simplifies to [43], its perspective counter part.

2. **RS**: Our proposed minimal 5-point rolling shutter pose and translational velocity solver as described in Section 4.2.2.

3. **RS + refinement**: Our proposed minimal 5-point rolling shutter pose and translational velocity solver with refined pose and velocities as described in Section 4.2.4.

The evaluations are done under varying image noise, increasing translational and angular velocities, and different shutter directions relative to the camera motions. There are a total of four different combinations for the shutter directions relative to the camera motions:

1. Horizontal shutter and sideway camera motion.
2. Horizontal shutter and forward camera motion.
3. Vertical shutter and sideway camera motion.
4. Vertical shutter and forward camera motion.

For each setting and method, we report the *median* error over 1000 random trials. Each trial consists of a random camera pose generated within the range of [0,1]m and [-0.01,0.01]rad for the respective axes, and the scene consists of 1000 randomly generated points with an average depth of 20m. We used an image resolution of 1000 pixels with a fixed focal length of 1000 pixels, which results in a field-of-view of about 53°. We assume a fixed rolling shutter scan time of 72ms for all experiments. The following error measure which averages the rotation and translation errors over all scanlines are used to evaluate the synthetic experiments:

- Angle difference in $\mathbf{R}$, averaged over all scanlines:

$$\delta_\theta = \frac{1}{N}\sum_n^N \cos^{-1}\left(\frac{Tr(\mathbf{R}_n\tilde{\mathbf{R}}_n^\top)-1}{2}\right), \tag{4.10}$$

- Translation difference, averaged over all scanlines:

$$\delta_t = \frac{1}{N}\sum_n^N ||\mathbf{t}_n - \tilde{\mathbf{t}}_n||, \tag{4.11}$$

where $\mathbf{R}_n$, $\mathbf{t}_n$ denote the ground-truth transformation for a given scanline n and $\tilde{\mathbf{R}}_n$, $\tilde{\mathbf{t}}_n$ are the corresponding estimated measurements and N represents the total number of scanlines.
Figure 4.3 and 4.4 show the average error plots from the three algorithms under increasing translational velocity, zero angular velocity and an image noise of 0.5 pixel standard deviation. It can be seen that our proposed method shows a constant error with increasing translational velocity, while the translational

and rotational errors for **GS + refinement** increased linearly. In Figure 4.5 and 4.6, we evaluate the robustness of the algorithm under varying pixel noise with a constant translational velocity of 6.9m/s and zero angular velocity. The results show that our proposed method is less sensitive to the image noise than the **GS + refinement** approach for all the four combinations of shutter directions and camera motions.

Figure 4.8 shows the error plots when the zero angular velocity assumption is violated. Here, we vary the angular velocity while maintaining a constant translational velocity of 6.9m/s and an image noise of standard deviation 0.5 pixel. Our proposed method with refinement **RS + refinement** is observed to be more robust in the angular velocity interval of $[0-2.2]$rad/s. It is important to note that in practice the angular velocity of a camera mounted on hand-held devices or a moving car is normally ≤ 2.2rad/s. This can be seen from our real-world data taken from a camera mounted on a moving car in Section 4.3.2. The car reached a maximum angular velocity of only 0.31rad/s when making a 90° turn. Motion blur might also occur for any angular velocity that is greater than 2.2rad/s, thus making it useless for pose estimation.

In general it might be counter intuitive that the algebraic minimization of the *RS* solution outperforms the **GS + Refinement** approach. The reason is the initial solution of the **GS** approach only finds a reduced set of inliers that satisfies the **GS** perspective model. This poorly distributed set of inliers does not constraint the camera motion well enough for the geometric refinement to converge to the correct solution. In our synthetic experiments the number of inliers obtained in the **GS** case drops by over 70% with a motion of 12m/s.

4.3.2 Streetview Data

We evaluate the proposed algorithm on 5 different datasets - Dataset (a)-(e). These datasets were captured by a Google Streetview car. The images have a native resolution of 1944×2592 pixels and are recorded at 4Hz. The shutter time for the rolling shutter camera is 72ms. This corresponds to a motion of 0.5m during image formation for a car driving at 25km/h. The camera poses are obtained from a GPS/INS system and interpolated to provide a position and orientation for each scanline. We will refer to these poses as the ground truth poses. It should be noted that the baseline between consecutive

Dataset	# of Cameras	# of 3D Points	Length (m)	Median δ_θ (rad)	Median δ_t (m)
(a)	118	215936	276.82	0.0014	0.0539
(b)	178	338806	377.44	0.0109	0.1698
(c)	190	326254	451.40	0.0050	0.2975
(d)	156	336712	353.61	0.0016	0.0423
(e)	162	310893	376.39	0.0030	0.0951

Table 4.1: Dataset overview

images taken from the rolling shutter camera is approximately 1m, and we are not using any temporal constraints to estimate the velocity and pose of the camera. An overview of the 5 sequences is given in Table 4.1. In the first step, we create a 3D map by using all the even numbered images from each of the sequences. We extract and match SIFT [68] features using [111]. The point correspondences are then radially undistorted and triangulated using the provided GPS/INS pose. 3D points with large reprojection error ($>$ 1 pixel) are discarded from the model. Figure 4.9 second row, shows the completed 3D models.

For each odd numbered image in the sequences, we extract SIFT features, radially undistort the keypoints and match them to the 3D model. This gives us the potential 2D-3D correspondences. The matches are used in RANSAC together with our 5-point minimal solver to find a consensus set. The pose and velocity hypothesis is refined by minimizing the objective function in Equation 6.5 with the Google Ceres [2] solver using all inliers. We compare the final pose to the ground truth using the same error measure as in the synthetic evaluation. The third and fourth rows of Figure 4.9 show the translational and angular error distributions for each of the datasets. We can see that both the translational and angular errors are very small (translational error ~ 0.05m and rotational error ~ 0.125rad) as compared to the ground truth. This shows the accuracy of our algorithm on real-world datasets.

It is also interesting to show that we could potentially use a rolling shutter camera mounted on a car as cheap velocity sensor to estimate the speed of the car from a single rolling shutter image. In Figure 4.10, we show two examples of the application on Dataset (d) and (e). The top row shows the ground truth velocities. The bottom row shows we have achieved a median error of only 2.01km/h and 3.4km/h for the speed estimate on the datasets.

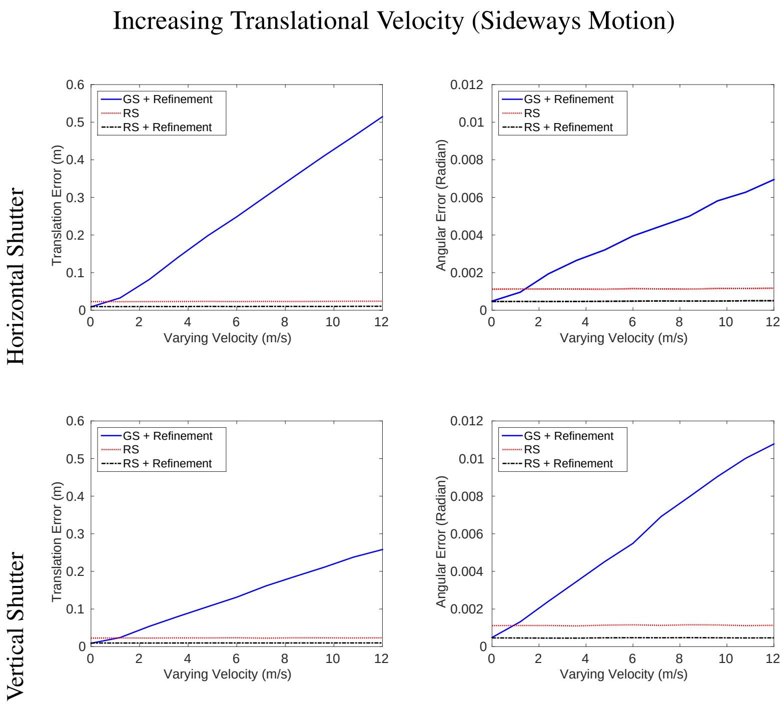


Figure 4.3: Evaluation on sideways motion, with increasing translational velocity with image noise of 0.5 pixel standard deviation and zero angular velocity. The first two columns show error plots for a sideways motions of the camera and the two last columns show errors for a forward (into the scene) moving camera.

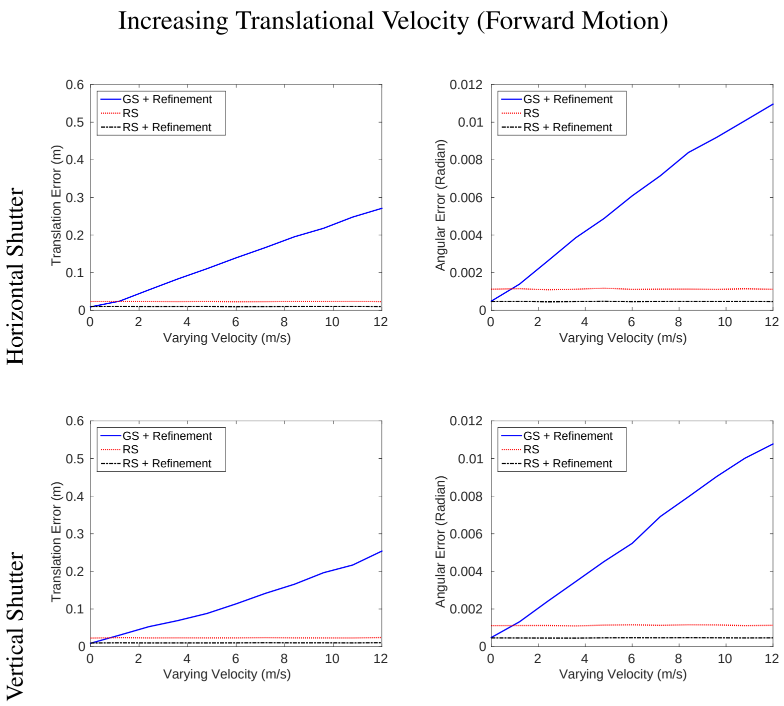


Figure 4.4: Evaluation on forward motion, with increasing translational velocity with image noise of 0.5 pixel standard deviation and zero angular velocity. The first two columns show error plots for a sideways motions of the camera and the two last columns show errors for a forward (into the scene) moving camera.

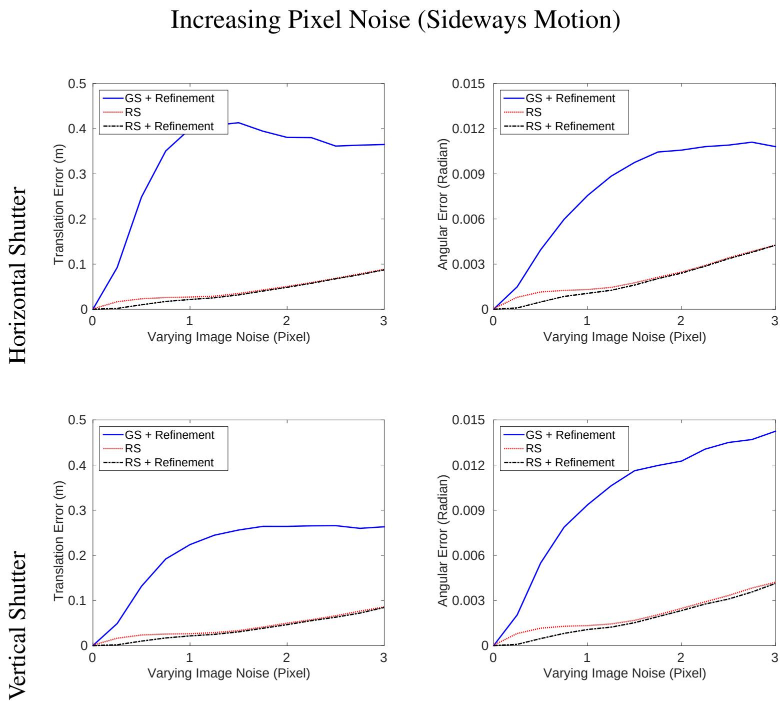


Figure 4.5: Evaluation on sideways motion, with increasing pixel noise with a fixed translational velocity of 6.9m/s and zero angular velocity. The first two columns show error plots for a sideways motions of the camera and the two last columns show errors for a forward (into the scene) moving camera.

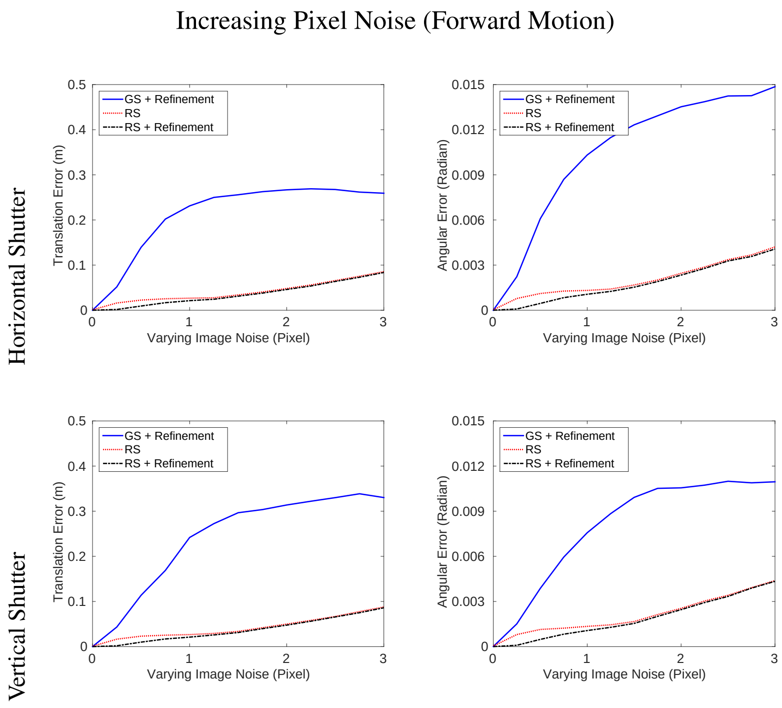


Figure 4.6: Evaluation on forward motion, with increasing pixel noise with a fixed translational velocity of 6.9m/s and zero angular velocity. The first two columns show error plots for a sideways motions of the camera and the two last columns show errors for a forward (into the scene) moving camera.

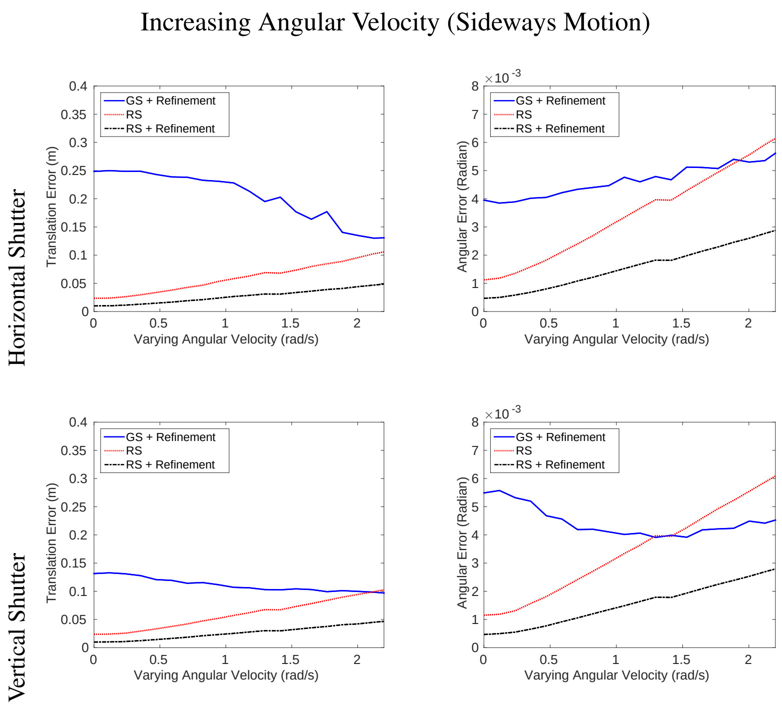


Figure 4.7: Evaluation on sideways motion, with increasing angular velocity with fixed image noise of 0.5 pixel standard deviation and translational velocity of $6.9m/s$. The first two columns show error plots for a sideways motions of the camera and the two last columns show errors for a forward (into the scene) moving camera.

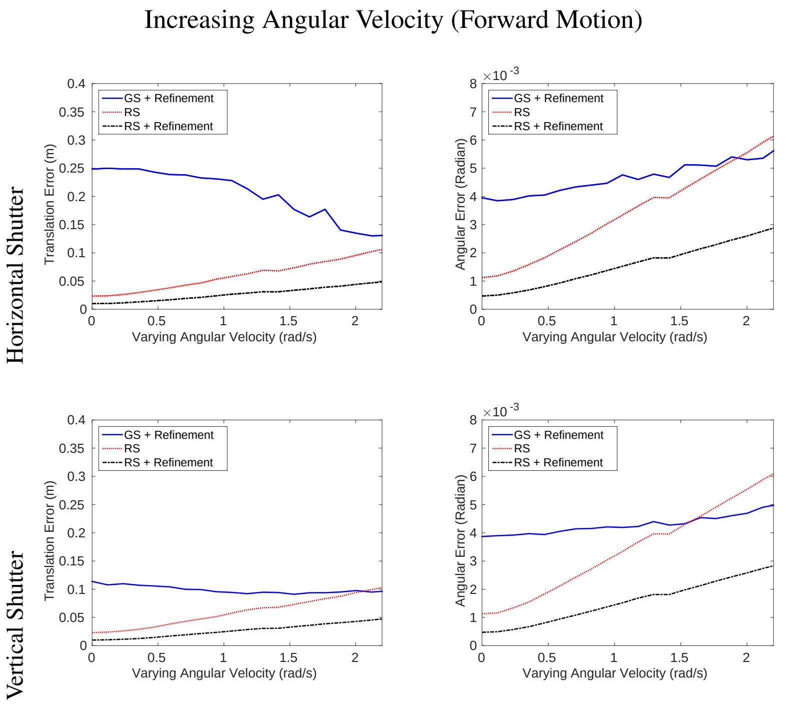


Figure 4.8: Evaluation on forward motion, with increasing angular velocity with fixed image noise of 0.5 pixel standard deviation and translational velocity of $6.9m/s$. The first two columns show error plots for a sideways motions of the camera and the two last columns show errors for a forward (into the scene) moving camera.

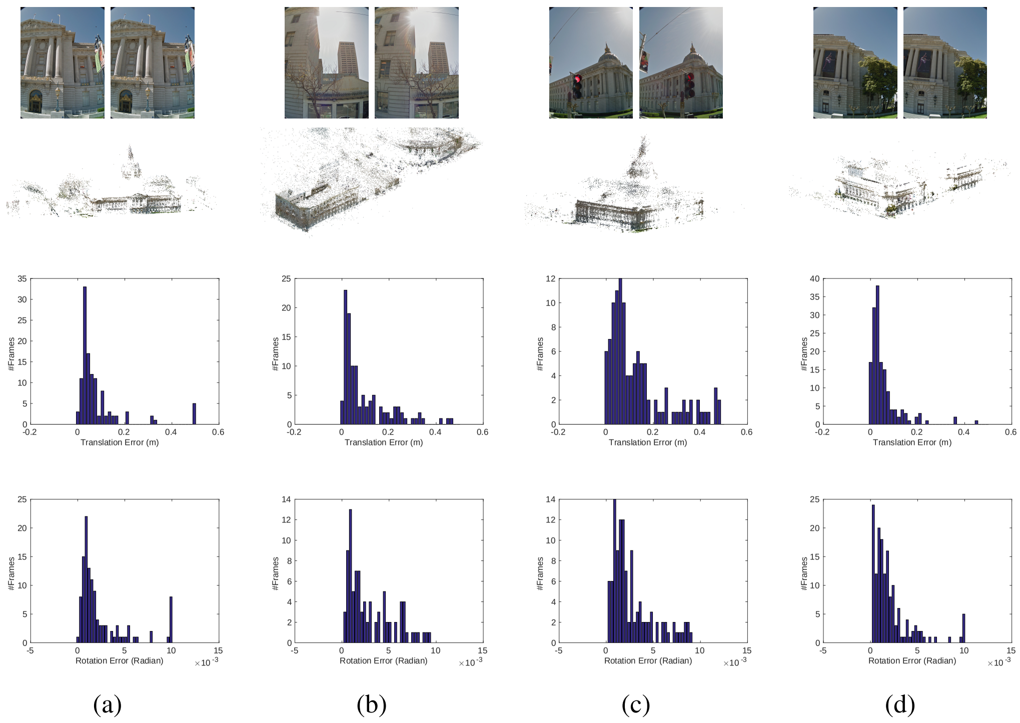
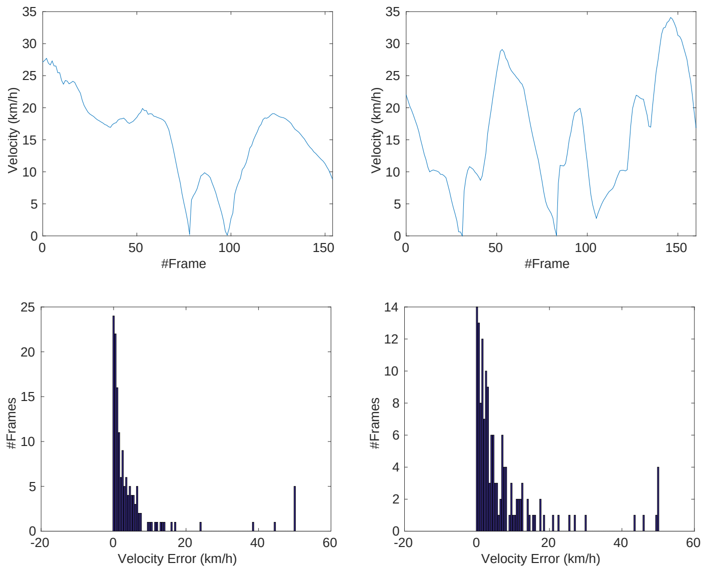


Figure 4.9: First row shows the sample input images for Dataset (a)-(d). The second row shows the sparse 3d reconstructions of the City Hall and surrounding buildings in San Francisco. The third and fourth rows show the distribution of rotation and translation error against ground truth. Results with fewer than 20 matches were removed. Note that the last bin of the histogram is expands to infinity.



Dataset (d) Dataset (e)

Figure 4.10: Top row ground truth velocity of the car. Bottom row error distribution of estimated car speed. In case of the Dataset (d) we achieve a median error of 2.01km/h and 3.4km/h for Dataset (e). Note that the last bin of the histogram is expanded to infinity.

5 Rolling Shutter Stereo

Contents

Visual 3D reconstruction of objects, scenes or whole cities nowadays seems to be a well understood problem, building on techniques like structure from motion and dense depth estimation (see e.g. [77, 85]). However, results published so far usually assume classical CCD cameras that capture images in a way that all pixels of the same image are being exposed at the same time. This is however not true for most CMOS sensors, such as those built into nowadays' smart phones or many industrial cameras [37]. Consequently, the analysis of rolling shutter cameras came into focus, where exposure of columns (scanlines) happens in sequential order leading to undesired distortion effects when the camera is not fixed during exposure. It has been shown recently that for hand-held smartphone cameras in static scenes, most of the rolling shutter effects can be compensated in the image (without 3D scene information), that is by compensating rotation [8, 27, 41, 88]. However, in case a high resolution camera is mounted on a moderately fast driving capture vehicle, strong rolling shutter effects will be introduced by the motion of the camera, even if the camera orientation is stable (similarly at a smaller scale, for video-based reconstruction of objects using a cell phone). Unfortunately, these effects depend on the distance to the objects, such that closer 3D points will be much more distorted than those very far away, making a simple 2D image warp into a global shutter image impossible. Also “standard stereo” rectification of image pairs (e.g. [34]) is in general not possible: Epipolar curve pairs, where each point on a curve in the left image maps to some point on a “corresponding” curve in the right image and vice versa, exist only in special configurations.

In this chapter we analyze the rolling shutter stereo problem and develop fast multi-view stereo algorithms that produce accurate 3D models from rolling shutter cameras. As real cameras often have lens distortion, and in particular those wide angle cameras often used for capturing streetlevel data, we also consider lens distortion, which we show makes the problem much more complex. To the best of our knowledge, no previous work exists on dense depth estimation with rolling shutter cameras and the common setting of lens distortion in a rolling shutter setting has not been analyzed. We therefore make the following novel contributions:

1. Practical discussion of fast-motion induced rolling shutter effects: Traditional stereo produces biased 3D results for standard streetlevel capture geometries

2. Analysis of interplay between rolling shutter and lens distortion: Correct undistortion requires 3D scene information.

3. Planar rolling shutter warp as a generalization of the plane induced homography

4. Multi-view stereo algorithm for rolling shutter cameras (with or without lens distortion)

In section 5.1 we will review previous work on rolling shutter cameras. We will then recapitulate the rolling shutter model and analyze fast motion and lens distortion effects in section 5.2. In section 5.3 we develop a warp for mapping a point of one rolling shutter image into another rolling shutter image, assuming a planar 3D scene. Based on this we then present both fast and accurate multi-view stereo algorithms in section 5.3.1. These are then evaluated quantitatively on textured laser scan models and qualitatively on real street-level data in section 5.4.

5.1 Related work

The chip-level architecture of a CMOS sensor and the reasons for the rolling shutter effect are described by Liang et al. [62] who also propose an optic-flow-like method to compensate rolling shutter effects for in-plane motion. Earlier, in [37] Geyer et al. had analyzed the effect of a rolling shutter camera, in particular for special camera motions and geometries (e.g. fronto-parallel, no forward components) and had suggested a scheme how to calibrate the shutter timings. They showed that in a very special setting a rolling shutter sensor behaves as a x-slits camera [117]. For those, Feldman et al. had discussed epipolar geometry [25]. Rolling shutter cameras are also related to pushbroom camera models [40] often used for satellite images (actually a special case [117] of the x-slits cameras), however for those, under straight motion, backprojected planes are parallel, while for rolling shutter cameras this does not hold.

Recently, several approaches for image stabilization for rolling shutter cameras have been proposed. Here, Bradley et al. [16] use stroboscope lighting and subframe warping to synchronize multiple rolling shutter cameras and to compensate the sequential exposure effects. Baker et al. [10] pose the rectification as a superresolution problem that can be solved using optical flow. Also Grundmann et al. [39] exploit local flow vectors to compensate rolling shutter for uncalibrated cameras, but using a mixture of homographies. In contrast, Hanning et al. [41] and Karpenko et al. [8] use gyroscopes of cell phones to compensate for rotational shake.

While the above approaches are rather 2D in nature, Forssen, Ringaby, Hedborg et al. applied structure from motion algorithms to tackle the problem for static scenes: First, Forssen and Ringaby [27, 88] had tracked features through cell phone video sequences and compensated cell phone rotation, which they identified as the dominant source of distortion for hand-held videos. In a later work, Hedborg et al. [45] have shown a full bundle adjustment including motion effects as well. Most recently Klingner et al. [51] proposed a structure from motion pipeline, for cameras mounted on a car, which uses relative pose prior along the vehicle path.

Our approach can be seen as the next step of a 3D reconstruction pipeline from rolling shutter cameras. Given camera motion and orientation (from

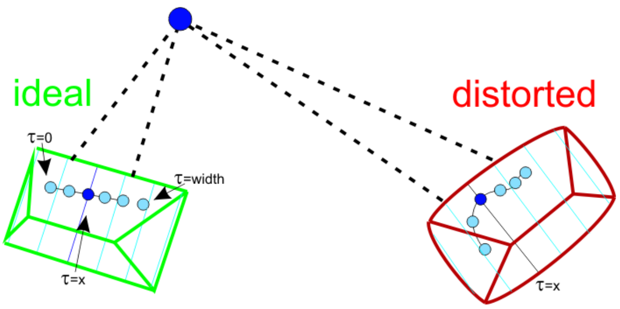


Figure 5.1: An ideal (green) and a distorted (red) camera observing a 3D point while moving straight. The projection of the point describes a straight line or a more complicated curve (depending on degree of distortion parameters). The distorted camera cannot be undistorted without depth information for the 3D point, as the time of exposure τ depends on the depth.

bundle adjustment and/or sensors), our goal is to densely estimate the 3D scene geometry from rolling shutter cameras. In particular, and in contrast to the work on hand-held cell phones, we consider the case where the camera undergoes fast motion (e.g. on a capture vehicle, or in a cell phone close to an object) introducing a depth-dependent rolling shutter effect. On top we consider lens distortion, which cannot be pre-rectified as that would change the image coordinate and thus the time when the particular 3D points was seen by the camera. However, using a plane-sweep stereo approach (see e.g. [116] or [48] for non pinhole cameras), we show how to solve depth estimation, lens undistortion and rolling shutter compensation at the same time. The approach is intended for motion stereo, i.e. with a single camera, which is however valid as well for moving camera rigs.

5.2 Rolling Shutter Camera Model

Lets look at the case where a 3D point $\mathbf{X}$ is observed by a global shutter pinhole camera P, i.e. it is projected to image position x given a known camera calibration matrix K (without loss of generality we assume K to be the identity matrix in the remainder of the chapter). In case of linear camera motion and constant orientation, the point moves on a straight line in the image, its position depending on the time τ:

$$\mathrm{x}_\tau \simeq \mathsf{P}_\tau \mathbf{X} = (R_0 \mid \boldsymbol{t}_0 + \tau \boldsymbol{t})\,\mathbf{X} \tag{5.1}$$

Now, we will move to the rolling shutter camera model, assuming that image column (scanline) r is exposed at time $\tau = mr + b$. For simplification of notation we assume $m = 1$ and $b = 0$, however, for a real system these coefficients need to be calibrated [37] and considered. In order to find out *when* $\mathbf{X}$ will be seen by the camera, we have to check, at which moment it is projected to an active scanline. For this we have to compute the x-coordinate (the scanline) of the projection into the image and take x_τ of Eq. 5.1 from projective space $\mathbb{P}^2$ to $\boldsymbol{x}_\tau$ into euclidean space $\mathbb{R}^2$

$$\boldsymbol{x}_\tau = \begin{pmatrix} (c_1\tau + c_2)/(c_5\tau + c_6) \\ (c_3\tau + c_4)/(c_5\tau + c_6) \end{pmatrix} \tag{5.2}$$

for some coefficients c_i depending on calibration, pose and 3D point. Then we look at the scanline (horizontal coordinate) that must match the time of exposure τ:

$$\mathrm{scanline}_{\boldsymbol{X},\mathsf{P}}(\tau) = (c_1\tau + c_2)/(c_5\tau + c_6) \overset{!}{=} \tau \tag{5.3}$$

Essentially, we are looking for the fixpoint of $\mathrm{scanline}(.)$ which leads to a quadratic equation in τ. The derivation was based on a straight simple motion model. In Tab. 5.1 we list a number of alternative parametric motion/camera models and the resulting degree of the τ polynomial (considering extra rotational or translational offsets is possible and will add more freedom to the motion patterns but will not change the degree of the polynomial). For each of those, to project a 3D point a polynomial in τ has to be solved to figure out whether the point is seen on the scanline that is currently exposed.

Many lenses, in particular wide angle lenses, show a significant amount of distortion and in the following we will briefly re-derive a standard radial/tangential distortion model that dates back to Brown [17]:

$$\boldsymbol{x}'_{\tau} = (1 + r_2 r^2 + r_4 r^4 + r_6 r^6)\boldsymbol{x}_{\tau} + \boldsymbol{dx} \tag{5.4}$$

with $\boldsymbol{x}_{\tau} = (x, y)^{\mathsf{T}}$ being the (undistorted) offset vector from the distortion center[1], $r = \|\boldsymbol{x}_{\tau}\|$, $\boldsymbol{x}'_{\tau}$ being the offset in the distorted image and

$$\boldsymbol{dx} = \begin{pmatrix} 2t_a xy + t_b(r^2 + 2x^2) \\ t_a(r^2 + 2y^2) + 2t_b xy \end{pmatrix}. \tag{5.5}$$

Herein, r_2, r_4, r_6, t_a, t_b are the distortion coefficients. In case radial or tangential distortion is present in the image, the curve $\boldsymbol{x}'_{\tau}$ described by a point in the image even under straight camera motion becomes more complicated and the degree of Eq. 5.3 will increase (see Tab. 5.1 and Fig. 5.1). Note that when the lens distortion of such a rolling shutter image is compensated (classical global shutter like inversion of Eq. 5.4), it means that straight lines in space will also become straight in the image, but that the shape of an original CMOS sensor scanline (those pixels that were exposed jointly at the same time) will become a more complicated curve rather than an image column. Consequently, the complexity is just shifted from the left hand side of Eq. 5.3 to the right hand side.

For short term motions (during exposure time of one image, which is usually a fraction of a second) of rolling shutter cameras the linear motion with no rotation (car driving straight) and the orbital motion (cell phone filming a handheld object) are the most important cases. The linear/linear case is somewhat more special and applies to panning cameras on a linear stage as those track-level "slow-motion" cameras used for the 100m sprints at the Olympic games. Also note that for the important cases the first (most significant) radial distortion coefficient can be considered for a closed form solution (polynomial degree up to 4).

[1]For simplicity of notation, we assume the distortion center at $(0, 0)^{\mathsf{T}}$

Motion	Orient.	Dist.	Pose P_τ	deg.
linear	const	no	$(\mathsf{I} \mid \tau\boldsymbol{t})$	2
orbital	linear	no	$\left(\mathsf{I} + \tau\,[\boldsymbol{r}]_\times \mid \boldsymbol{t}\right)$	2
spiral	linear	no	$\left(\mathsf{I} + \tau\,[\boldsymbol{r}]_\times \mid \tau\boldsymbol{t}\right)$	2
linear	linear	no	$\left(\mathsf{I} + \tau\,[\boldsymbol{r}]_\times \mid (\mathsf{I} + \tau\,[\boldsymbol{r}]_\times)\tau\boldsymbol{t}\right)$	3
linear	const	r_2	$(\mathsf{I} \mid \tau\boldsymbol{t})$	4
orbital	linear	r_2	$\left(\mathsf{I} + \tau\,[\boldsymbol{r}]_\times \mid \boldsymbol{t}\right)$	4
spiral	linear	r_2	$\left(\mathsf{I} + \tau\,[\boldsymbol{r}]_\times \mid \tau\boldsymbol{t}\right)$	4
linear	linear	r_2	$\left(\mathsf{I} + \tau\,[\boldsymbol{r}]_\times \mid (\mathsf{I} + \tau\,[\boldsymbol{r}]_\times)\tau\boldsymbol{t}\right)$	5
not const.	any	r_2, r_4 r_6 t_a, t_b		≥ 8

Table 5.1: Some common short term motions and the resulting polynomial degree when (not) considering distortion for obtaining the τ in a rolling shutter camera (see also [37]). Note that for very short term (intraframe) motion on fast driving cars we can assume constant orientation but that the more general models would not lead to a significantly more difficult problem.

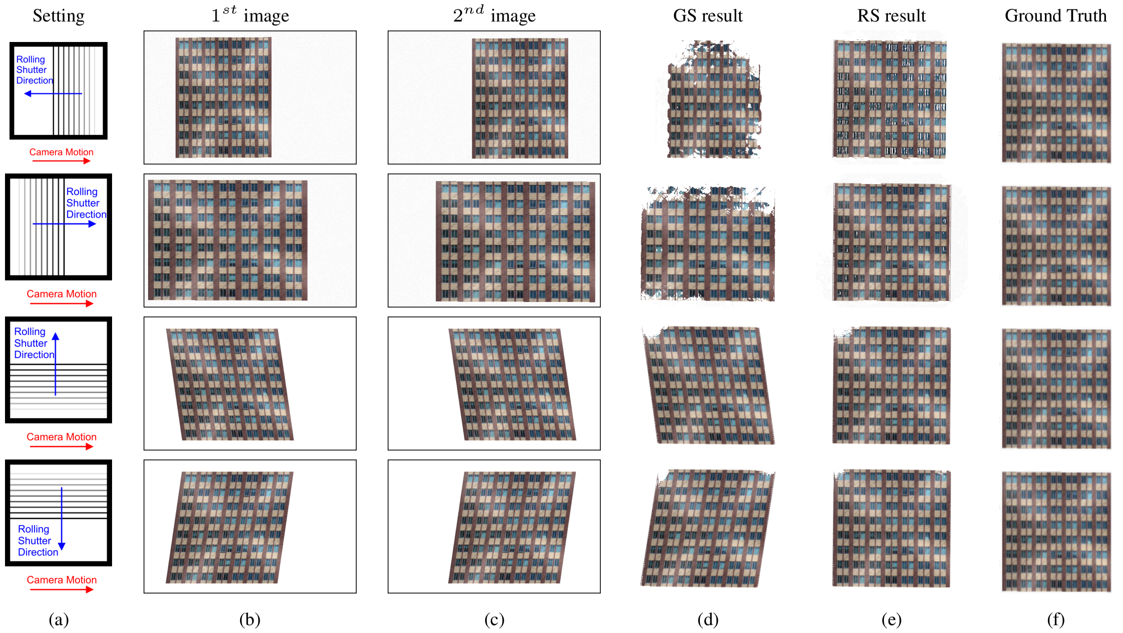


Figure 5.2: Reconstruction of a square building facade observed by a rolling shutter camera moving parallel to the facade from left to right, with different simulated rolling shutter directions. (a) shows rolling shutter direction relative to motion direction, resulting in some image (b) and later in another image (c). Column (d) shows the reconstruction when assuming the same pose for all pixels (classical global shutter model), where one can see horizontal stretch, horizontal compression or slant in the 3D model. Finally, column (e) shows the reconstruction using our formulation, taking into account the rolling shutter effect, which matches the ground truth 3D model (f).

5.2.1 Rolling Shutter Observability

Rolling shutter needs to be considered only when its effects are significant, i.e. for stereo in the range of one pixel or more. In the following we concretize the assessment of [37] with practical numbers and considerations to allow for a decision of whether a rolling shutter model makes sense for a particular capture configuration. We assume that a capture vehicles drives at a certain speed v (e.g. 25 km/h) and uses a camera with a certain field of view ϕ (e.g. 90 degrees) and image width w (e.g. 2000 pixels). We assume that all lines of an image have been exposed after t (e.g. 72 ms), i.e. there is a time difference of $t/2$ between the center scanline and a boundary scanline of an image. The camera position error compared to the center pose is $\Delta x = t/2 \cdot v$. The focal length can be computed as $f = \frac{w}{2}/tan(\frac{\phi}{2})$. Looking at some point on the optical axis of the camera (i.e. $(0\ 0\ z)^\mathsf{T}$ in camera coordinates), it will be projected to the principal point (in a global shutter camera model). If that camera now moves by some amount Δx depending on the speed defined above, then if we want at most 1 pixel displacement, the 3D point must be at least $z > f\Delta x$ away, i.e.

$$z > \frac{\frac{w}{2}}{tan(\frac{\phi}{2})} \cdot \frac{t}{2} \cdot v \tag{5.6}$$

which, inserting the above values and approximating $tan(x) \approx x \cdot tan(45°)$, leads to the rule of thumb

$$z_{\text{min}} \approx 6.25m \frac{w_{\text{[pixel]}}}{\phi_{[degree]}} \cdot t_{\text{[sec]}} \cdot v_{[km/h]} = 250m \tag{5.7}$$

That is, for ten times less resolution or ten times less speed there is still at least one pixel error up to $25m$ distance but neither such speeds nor such resolution is any useful. Because *locally* these errors are not as visible (between neighboring scanlines the error is 1000 times smaller, as there are thousand scanlines between the center and the border) they might seem to be of minor importance, however as can be seen above *for accurate reconstruction from a driving car they are significant.*

Vertical or Horizontal Rolling Shutter? For vehicle mounted cameras there are several considerations for how to mount the camera, such as different field of view in x and y direction, mounting space with respect to other cameras, full dome coverage and so on. Besides those, the direction of the rolling shutter plays an important role. For camera planes parallel with the facades and rolling shutter orthogonal to the motion direction, a shearing effect will be visible in each image (maybe less visually pleasant when displayed as raw image) and such images do not align well with Manhattan structures in the scene. On the other hand, when the rolling shutter is parallel with the motion, the image will be shrunk or stretched in that direction and for certain driving speeds undersampling issues may appear. The resulting images and qualitative effects when observing a plane with different shutter directions can be seen in Fig. 5.2.

Independently of the direction of the rolling shutter, we will develop a depth dependent rolling shutter image warp in the next section, that can warp one rolling shutter image into another one taken from somewhere else, assuming some scene plane Π, similar in spirit to a plane-induced homography (with the goal of enabling plane sweep stereo).

5.3 Rolling Shutter Warp Across a Plane

To warp a point from one rolling shutter camera to another, we first backproject it to a plane Π and then project it into the other image.

RS Backprojection of pixel onto space plane Π: Given some pixel position $\mathbf{p} \in \mathbb{P}^2$ in a rolling shutter image, from its scanline we know immediately the time of exposure τ_p and consequently the corresponding projection matrix P_{τ_p}. Consequently, we can choose a 3D point $\mathbf{X}_i \in \mathbb{P}^3$ on the ray through the camera center $\mathbf{C}_{\tau_p} \in \mathbb{P}^3$ that projects to $\mathbf{p}$. All points $\mathbf{L}_i$ on that ray can be represented as

$$\mathbf{L}_i = \mathbf{C}_{\tau_p} + \lambda \mathbf{X}_i, \qquad \lambda \in \mathbb{R} \tag{5.8}$$

All points $\mathbf{X}$ that lie on the plane $\Pi \in \mathbb{P}^3$ fulfill

$$\Pi^{\mathsf{T}}\mathbf{X} = 0, \quad \text{where} \quad \Pi = (\boldsymbol{n}_\Pi \;\; -d)^{\mathsf{T}}, \tag{5.9}$$

and substituting Eq. 5.8 into Eq. 5.9 we arrive at a linear equation in λ

$$\Pi^{\mathsf{T}}(\mathbf{C}_i + \lambda \mathbf{X}_i) = 0, \tag{5.10}$$

that allows to find the 3D intersection $\mathbf{X}_\Pi$ of the plane Π and the backprojected ray.

RS Projection of plane point into other view: The time of exposure of a certain 3D point can be computed according to Eq. 5.3, that is quadratic in τ or, in case of distortion, using $\boldsymbol{x}'_\tau$ from Eq. 5.4 substituted for $\boldsymbol{x}_\tau$ in Eq. 5.3:

$$\frac{\alpha_7 \tau_q^7 + \alpha_6 \tau_q^6 + \cdots + \alpha_0}{\beta_7 \tau_q^7 + \beta_6 \tau_q^6 + \cdots + \beta_0} = \tau \tag{5.11}$$

that can be rewritten as

$$\gamma_8 \tau_q^8 + \gamma_7 \tau_q^7 + \cdots + \gamma_0 = 0, \tag{5.12}$$

for some $\alpha_j, \beta_k, \gamma_l \in \mathbb{R}$. The degree of the polynomial depends on the lens distortion and the motion model as can be seen in Tab. 5.1. Up to fourth order, i.e. using only the first radial distortion coefficient and one of the important motion models, this can be solved in closed form for the time of exposure τ_q in the other image. Only τ_qs are valid that lie in the exposure time interval, in our case $[0; \mathrm{width} - 1]$. In the rare case that more than one solution fulfills this, the same 3D point is seen multiple times in the same image (remember the rolling shutter creates a multi-perspective image when the camera moves). If we just want to find the color of the point (as is the case for our warp), all solutions are valid and we simply choose the earliest time of exposure τ_q. Given τ_q, we know the camera pose and we can project the 3D point on the plane to finally obtain the image coordinates $\mathbf{q}$ in the other image, which completes the warp.

For the full radial/tangential distortion model according to Eq. 5.4, we obtain a polynomial that cannot be solved in closed form. Consequently, we perform gradient descent on Eq. 5.12, initialized with $\tau_0 = \mathrm{width}/2$.

Although the previously described warp can be fully parallelized and runs on the GPU it is computationally expensive since it requires to solve for the time τ for each pixel individually. We suggest - and later evaluate - two approximate strategies, that promise a speedup at the cost of some accuracy:

Fast approximation 1 (FA1): global shutter lens undistortion An efficient approximation is to perform the expensive lens undistortion globally, and then solve for the quadratic Eq. 5.3 in closed form. This is in particular useful when the lens distortion is minor, because then the time of exposure does not change much with or without distortion. In this case the undistortion can be precomputed offline using a lookup table (as standard for global shutter undistortion) and has to be done only once per image (if warps across multiple planes are run as in plane sweeping there is no need to run it per plane).

Fast approximation 2 (FA2): coarse grid computation of warp's texture coordinates Alternatively, rather than computing τ and then the resulting texture coordinate for each pixel, we propose to evaluate the texture coordinates in dependence of τ on a coarser grid (e.g. only every 10 pixels) which is then used to compute the actual texture lookup coordinates using texture interpolation. This approach (FA 2) can exploit highly optimized GPU texture handling.

The speedup of the approaches above is given in Tab. 5.2. The timings are evaluated on a GeForce GTX 680 graphics processing unit.

5.3.1 Integration to Plane Sweep Stereo

For global shutter cameras, it has been shown that the ability of the graphics processing unit to handle smooth warps [116] can be exploited for real-time stereo approaches. Having understood under what speeds, resolutions and distances a rolling shutter camera model must be used we can exploit the warps of the previous section in a plane sweep approach where we hypothesize a scene plane, warp our image across that scene plane into another reference view and determine the agreement of those images for each pixel. This is repeated for a number of planes to obtain a whole cost volume, i.e. we obtain costs (dissimilarity) for each plane hypothesis at each pixel position of the reference view. In order to robustify the approach with respect to partial occlusions, we generate the cost volume in the reference view from n neighboring views and for each plane at each pixel consider only the mean of the k best correlation costs out of the n neighbouring views.

On that cost volume, smoothness terms can be used and any suitable optimization technique to solve them. We follow smoothness terms and optimization strategy as proposed by Hirschmüller [47]. The result is a depth map for the reference view encoding a depth value for each pixel. Using also the image and the camera poses this can be used to generate a 3D model.

5.4 Evaluation

We evaluated the proposed rolling shutter stereo algorithm on both synthetic and real datasets that mimic a single rolling shutter camera mounted on a moving car (allowing motion stereo to be computed from consecutive images). All synthetic data are available on the project website [2].

Implementation Details: For the evaluations in this chapter we stick to a simple plane sweep model with a single plane normal (e.g. obtained from dominant scene planes [36] in a prior sparse reconstruction step like [45] or [51]) and a single sweeping direction. The sweep is performed by creating additional planes within the distance range $[D_{min}, D_{max}]$. The planes are sampled approximately linearly in image space, such that a warp over two neighbouring planes results in a pixel displacement of maximum one pixel distance. A warp over a plane is computed by first undistorting a pixel and intersecting the ray passing through the pixel with the plane being considered. The intersection point is then projected into the reference view according to the formulation presented in section 5.3. The dissimilarity measure used is 1-NCC (normalized cross correlation) on a 5×5 window. However, the similarity is summed up over multiple pyramid levels [116], giving always 1/4 weight to the smaller pyramid level. For choosing the k best views we choose $k = 3$ out of $n = 7$, however for the synthetic experiments just two views have been used. Finally, for each pixel a geometric verification step is performed once the depth map for the next reference view has been computed: Each pixel of depth map one is backprojected into space, projected into the other

[2]http://cvg.ethz.ch/research/rolling-shutter-stereo

image and compared to the depth estimated for that position. Discrepancies of more than 0.1m result in the depth value being declared invalid.

5.4.1 Ground Truth Evaluation

Rolling Shutter Direction: First, we qualitatively analyze the effects, when ignoring rolling shutter in stereo algorithms for different shutter directions: In Fig. 5.2 we texture a square plane in 3D space and synthetically generate two rolling shutter images. Already the shape of the images looks very different (squeezed, stretched, slanted). Consequently, when ignoring this effect and performing standard stereo, the reconstructions are also biased. For this setting we chose quite strong motion to visualize the effects, but it should be clear that the same type of systematic errors will appear also at smaller speeds. Note that the obtained (biased!) depths maps for global shutter are dense; This means that obtaining visually plausible results when applying a global shutter model to rolling shutter data does not mean the data is actually correct.

Quantitative Evaluation using Ground Truth: The datasets *castle* and *old town* were originally captured using a 3D laser scanner. The resulting point clouds have been smoothed, meshed and textured with high resolution photos. We then define a plausible streetlevel camera path and render 976 images (image resolution 976×732) along the path. From each of those we pick one column and compose a novel image out of these scanlines to simulate the rolling shutter effect (afterwards, the GPU's z-buffer is handled in the same way to obtain a rolling shutter depth map). This allows for evaluation of absolute 3D errors of the 3D reconstruction algorithms in meters using extremely realistic geometry and texture. For this setup the camera is assumed to have a linear motion with a constant orientation (first motion model). We evaluated rolling shutter stereo against global shutter stereo while the camera undergoes a motion of 0m, 0.318m, 0.636m and 1.27m respectively (castle) and 0m, 0.122m, 0.243m, 0.487m (old town) during exposure time, simulating different speeds. The baseline lengths between the two images were 3.9m and 0.75m respectively. This corresponds to maximum driving speeds of 65km/h

	speed / warp [ms]	median [m]	MAD [m]	fill rate
FA 1	10.0	40.88 / 49.51	3.242 / 3.17	40.8% / 32.9%
FA 2	2.2	1.02 / 0.26	1.02 / 0.22	52.8% / 58.6%
RS	27.7	0.041 / 0.085	0.032 / 0.077	76.3% / 62%

Table 5.2: Evaluation of the different warps, speed vs. accuracy on the castle and old town dataset. We use a grid resolution of 1/10 of the image resolution (976×732) for FA2.

(castle) and 24km/h (old town) for exposure time of approximately 1/14s (as in our real system).

In Fig. 5.4 and Fig. 5.3 it can be seen that the global shutter algorithm performs worse with increasing rolling shutter effect while the rolling shutter algorithm is approximately constant. We visualize the 3D error, that is the distance between the estimated 3D point and the GT 3D point. Note that the GS algorithm shows errors of more than a meter which were not detected by the final 0.1m depth consistency check, confirming again that the errors when using the global shutter model are significant but hard to detect.

Approximation 1 and 2 perform in between in terms of quality, however they have a completely different error pattern. While FA1 shows a global, systematic error because of the incorrect lens undistortion, FA2 performs correct at the grid, however shows a high frequency error that increases inside the grid cells.

General motion: In another experiment (see Fig. 5.5) we construct the rolling shutter images in a way that the motions during exposure of image 1 is in a different local direction than the one of image 2. This happens when using different rolling shutter cameras on a car which are looking into different directions. In this case there is no longer just a systematic bias in the data, but global shutter stereo just cannot find the correct correspondence any more.

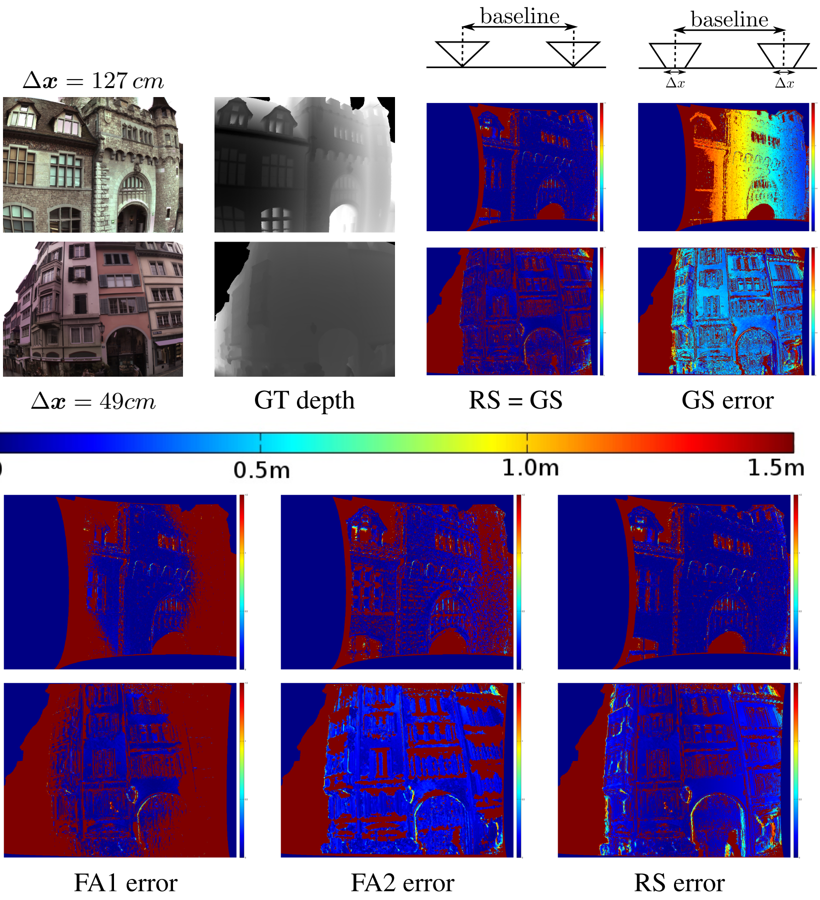


Figure 5.3: 3D error visualization for rolling shutter image pairs of two ground truth scenes (top and bottom): For no motion during exposure ($\Delta \boldsymbol{x} = 0$, that is global shutter) all algorithms produce the same results. For the maximum motion according to Fig. 5.4. For the global shutter, the error is generally higher and produces a systematic offset depending on depth and also distance to the distortion center.

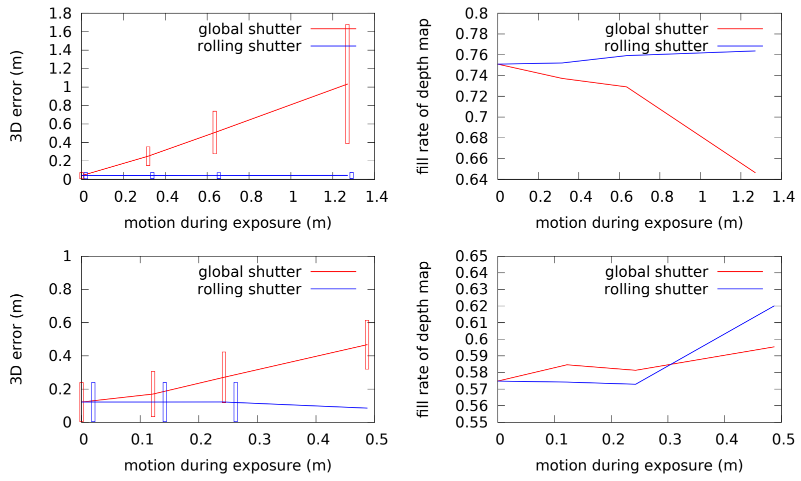


Figure 5.4: Top: castle sequence, bottom historic town center. Left: Median 3D error (boxes indicate median absolute deviation) when using a global or a rolling shutter algorithm. Right: corresponding fill rates of depth maps.

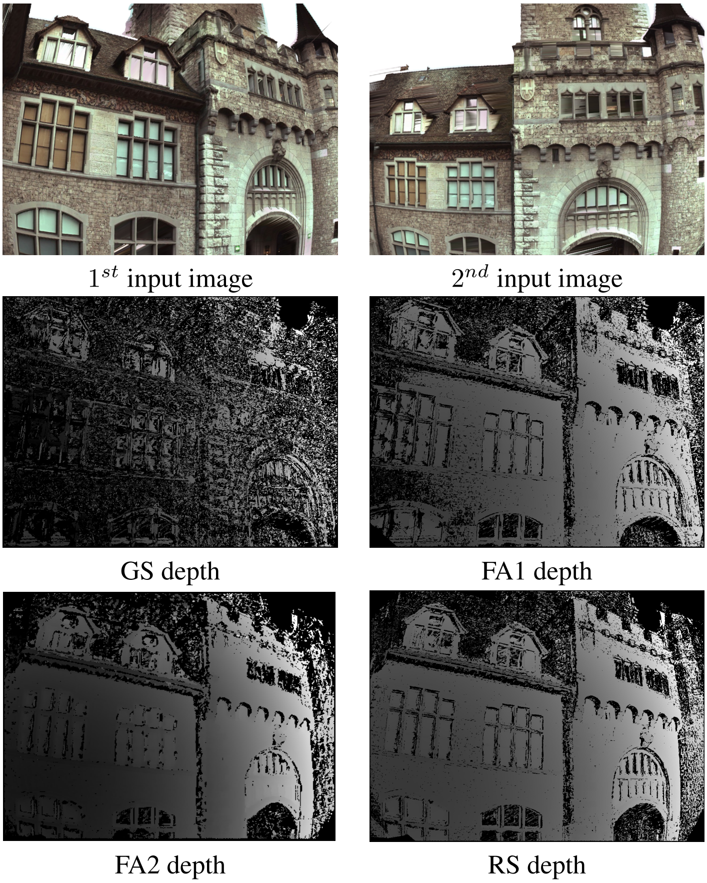


Figure 5.5: Top row, input images captured with independent linear motions (sideways- and forward-motion). Global shutter (GS) stereo fails as the correct correspondence is not in the search range. Fast approximation 1 (FA1) gives consistent depth which degrades towards the image boundaries. Fast approximation 2 (FA2) provides consistent depth at the grid vertices which then degrades inside the grid cell. Rolling shutter (RS) stereo provides throughout consistent depth.

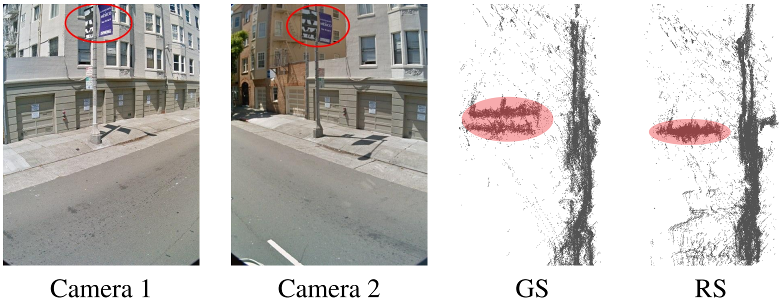


Figure 5.6: Left, sample input image of two cameras mounted on a car. We apply GS and RS stereo on both camera streams independently and fuse the resulting models into a single coordinate frame. Right, bird's eye view of GS and RS reconstruction. Note how the pole aligns in the RS reconstruction, while in the GS reconstruction it appears as two different poles.

5.4.2 Evaluation on Real Data

Real data has been recorded using a capture vehicle driving at different speeds. The exposure time of the camera was 72ms (approximately 1/14 s). For the *Oak street* there is approximately 0.5m displacement during exposure and a baseline of 2m between frames, driving speed was 25km/h. For the *Fillmore street* displacement was 0.74m, baseline 2.6m and driving speed 37km/h, with an image resolution of 1944×2599, results are given in Fig. 5.7. We compare GS and RS stereo on the Oake street sequence and observe that the facade is reconstructed further away from the camera in the GS case, compared to the RS case. Objects reconstructed from different cameras independently and fused into a single coordinate system don't overlap in the GS case while they do in the RS case, see Fig. 5.6.

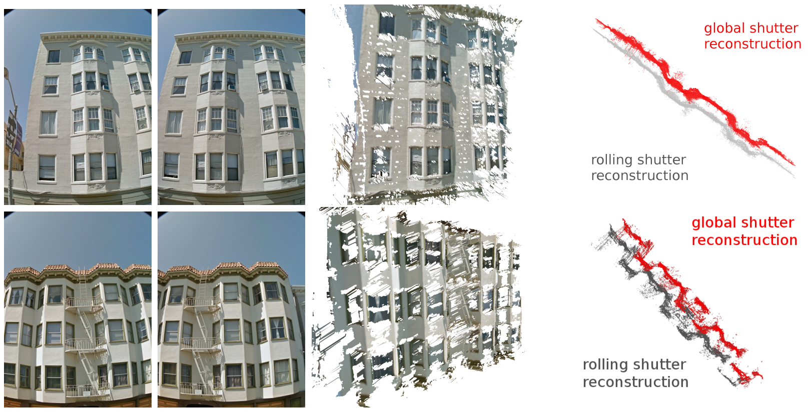

global shutter reconstruction
rolling shutter reconstruction
global shutter reconstruction
rolling shutter reconstruction

Figure 5.7: Real world reconstruction of the Oak street (top) and Fillmore street (bottom) data sets show systematic differences in bird's eye view (right column).

6 Sparse to Dense 3D Reconstruction from Rolling Shutter Images

Contents

The majority of image sensors on the market today (found in mobile phones, and compact cameras etc) are CMOS sensors. In contrast to standard CCD sensors which have global sensor readout, classical CMOS sensors have sequential readout. This leads to sequential exposures of each image row or column - commonly known as rolling shutter. For example, typical readout times in todays mobile phones are around 10-40ms [83, 104]. Given such a delay, significant deformations appear in the image when the camera is in motion. Assuming a camera with a readout time of 35ms and moving at 5km/h, objects which are closer than 25m will show deformations due to the rolling shutter [92] effects.

In the last decade, Structure from Motion (SfM) techniques have been used to build 3D models from both ordered and unordered image sequences [1, 28, 61, 106]. These SfM techniques rely on a global shutter camera model and become brittle when used on rolling shutter images [64, 67]. Hedbors [45] introduced a bundle adjustment method which provides accurate structure and motion from a rolling shutter video stream by using a continuous time parametrization of the camera pose. It is however limited to continuous image streams with small baselines and does not work on sparse images with wide baselines. Moreover, none of the existing works showed a full pipeline that does full sparse to dense 3D reconstruction from rolling shutter images with wide baselines.

In this chapter, we propose and implement a pipeline for sparse to dense 3D construction with wide baseline images captured from a fast moving rolling shutter camera. Wide baseline images are captured at a low frame rate of 4Hz with our rolling shutter camera mounted on a moving car moving at 17km/h. Specifically, we propose a cost function for bundle adjustment that models the rolling shutter effect, incorporates GPS/INS readings, and enforces pairwise smoothness between neighboring poses. We optimize over the 3D structures, camera poses and velocities. In contrast to [59] that minimizes the absolute difference of the GPS/INS and camera poses, our smoothness term minimizes the difference between neighboring camera relative poses and their corresponding neighboring GPS/INS relative poses. As a result, our bundle adjustment is able to compensate for the drifts that are accumulated in large scenes from the weak visual connectivity of wide baseline images without the risk of causing discontinuities in the poses.

The optimized camera poses are used to compute dense motion stereo. We adopt the plane sweep algorithm for rolling shutter camera proposed by Saurer $et\ al.$ [92]. We show that we can achieve a $7\times$ speed up in the depth map computation without losing accuracy with our novel interpolation scheme for the rolling shutter plane sweep stereo algorithm. We evaluate and show the feasibility of our approach on a 2.6km sequence, and compare our sparse and dense 3D reconstructions to those obtained from global shutter reconstructions.

Our contributions in this chapter can be summarized as follow:

- Propose and show a working pipeline for full sparse to dense 3D reconstruction from large scale wide baseline rolling shutter images.
- New cost function for bundle adjustment that models the rolling shutter

effect, incorporates GPS/INS readings, and enforces pairwise smoothness between neighboring poses.

- Achieved 7$\times$ speed up in depth map computation without losing accuracy with our novel interpolation scheme on the 3D planes for the rolling shutter plane sweep stereo algorithm.

6.1 Related Work

In the recent years, an increasing number of standard 3D vision algorithms originally designed for global shutter cameras are reformulated to include the rolling shutter camera model. These algorithms cover many aspects of 3D vision from camera calibration, pose estimation, bundle adjustment to dense motion stereo etc. However, none of these existing works showed a full sparse to dense 3D reconstruction from wide baseline rolling shutter images.

The authors of [37, 83] suggested algorithms for the calibration of rolling shutter timing. Others [8, 27, 41, 62, 88] have looked into rolling shutter wobble corrections using additional sensors such as gyroscopes or assuming that a rolling shutter wobble is induced by only a pure rotational motion. [7, 93] addressed the rolling shutter pose estimation problem with the minimal solutions from different variants of rolling shutter camera model. Meilland *et al.* [73] proposed a RGB-D SfM algorithm which simultaneously solves for motion blur and rolling shutter deformations. In [70], the authors proposed an iterative algorithm to solve for the camera pose and velocity.

A rolling shutter camera bundle adjustment for a continuous stream of small baseline images was shown in [45]. They reformulate the bundle adjustment problem such that it requires 6 additional parameters compared to the global shutter version. These 6 additional parameters are due to the exploitation of the fact that the pose of the last scanline corresponds to the pose of the first scanline in the next frame. In [51], the authors modelled a rolling shutter camera rig as a generalized camera, where relative poses between cameras are obtained from a GPS/INS system. In the bundle adjustment, they optimized for one global camera rig pose, while using a rolling shutter aware reprojection error.

Ait-Aider *et al.* [6] proposed a stereo algorithm for a rolling shutter stereo rig. Given a set of point correspondences, they recover the object pose and velocity by optimizing a non-linear system of equations. A challenge of finding pixel correspondences between image pairs arises in the monocular setup. While this is done by searching along the epipolar line for global shutter stereo, the search becomes difficult for the rolling shutter stereo where the pixel correspondence lies on an epipolar curve. The curvature of the epipolar curve depends on the motion and lens distortion of both cameras. In [92], the authors addressed this problem with a monocular rolling shutter plane sweep stereo algorithm. The main drawback of the algorithm is that it is computationally expensive to solve a high order polynomial for each pixel at various depths. We built upon their findings and proposed a novel algorithm which is $7\times$ faster in speed and provides similar accuracy in depth estimation.

6.2 Reconstruction Pipeline

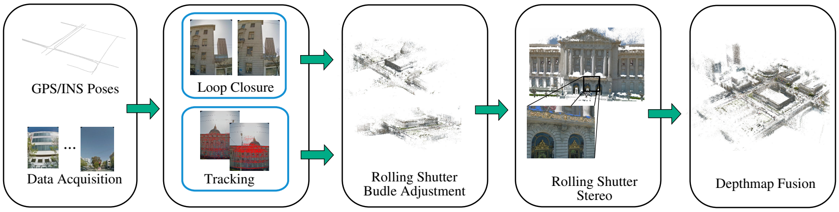


Figure 6.1: Overview of the reconstruction pipeline.

Fig. 6.1 illustrates our proposed pipeline for sparse to dense 3D reconstruction from rolling shutter images. Our proposed reconstruction pipeline follows the standard SfM approach [84] with modifications made for wide baseline rolling shutter images. In detail:

Data Acquisition: Images are captured with a rolling shutter camera rig mounted on a car. The rig consists of 15 cameras arranged such that they cover over 80% of a sphere. In this work, we only consider 8 cameras which point sideways to the driving direction. Images from consecutive frames have wide baseline because our cameras are recording at a low frame rate of 4Hz. Each image is also synchronized with a GPS/INS position.

Tracking: We start by extracting SIFT [68] features using [111] on the radially distorted images. Features are then matched between neighbouring images using a GPU brute-force matcher creating tracks between consecutive frames. Frames which show little parallax (< 50 pixel) are removed from the tracks.

Loop closure: Loop closures are detected by comparing the GPS/INS poses. Images that are within a radius of 15m are considered as potential loop closures. Potential loop closures are verified geometrically using a rolling shutter pose estimation algorithm similar to [93]. We consider a loop closure to valid if it is within the vicinity of their original GPS/INS pose (within 15m) and there is enough inliers (> 100) from the geometric verification. Each detected loop closure is added to bundle adjustment in the form of a pose graph as an additional constraint (see 6.3 for more details).

Rolling Shutter Bundle Adjustment: Tracks are first radially undistorted with the standard radial/tangential distortion model proposed by Brown [17]. Next, the undistorted keypoints are triangulated with the camera poses provided by the GPS/INS system. It is important to note that the camera poses provided by the GPS/INS system are not perfect due to systematic errors in calibration and multi-path problems in the urban environment. We use the GPS/INS system for a rough pose estimate of each scanline in the image. We then use our proposed rolling shutter aware bundle adjustment to optimize over the 3D points, and camera extrinsics i.e. poses and velocities, while enforcing smoothness constraints between neighbouring poses in the pose graph. The bundle adjustment refinement process is further discussed in Section 6.3.

Rolling Shutter Stereo: The refined poses are then used to compute a dense 3D model using a multiview and multi-resolution rolling shutter plane sweeping stereo algorithm similar to [92]. Here, we show that our novel interpolation scheme on the 3D planes for the rolling shutter plane sweep stereo algorithm allows us to achieve a $7\times$ speed up in the depth map computations for dense reconstruction without losing accuracy. The depthmaps are regularized using Semi-Global matching [47]. More details are given in Section 6.4.

Depthmap Fusion: Finally, all 3D models are merged into a single coordinate frame. Only depth values which obtained support from at least 3 or more views are kept and used to render a dense point cloud.

6.3 Rolling Shutter Bundle Adjustment

Traditional bundle adjustment [106] minimizes the re-projection error formulated as:

$$\underset{\mathbf{R},\mathbf{t}}{\operatorname{argmin}} \sum_m \sum_n ||\mathbf{x}_{m,n} - \mathbf{K} \cdot \mathbf{D}(\pi(\mathbf{P}_m, \mathbf{X}_n))||^2, \tag{6.1}$$

where $\mathbf{K}$ is the camera intrinsics parameter, $\mathbf{D}$ is the radial distortion function, $\mathbf{P} = [\mathbf{R}, -\mathbf{R}\mathbf{t}]$ is the camera extrinsics, i.e. rotation and translation, $\mathbf{x}_{m,n}$ is the feature point corresponding to the 3D point $\mathbf{X}_n$ observed by the camera m, and $\pi(.) : \mathbb{P}^3 \to \mathbb{P}^2$ denotes the projection function. This formulation assumes a global shutter camera model. For a moving rolling shutter camera, each scanline gets exposed at a different place in space along the motion trajectory. As a result, Eq. 6.1 no longer holds.

Similar to [7,37] we propose to use a constant translational and rotational velocity parametrization for the camera pose:

$$\mathbf{t}(\tau) = \mathbf{t}_0 + \mathbf{v}\tau, \tag{6.2a}$$

$$\mathbf{R}(\tau) = \mathbf{exp}(\Omega\tau)\mathbf{R}_0, \tag{6.2b}$$

where $\mathbf{v}$ denotes the translational velocity, Ω denotes the angular velocity and τ the time of exposure (time delay between first an current scanline). $\mathbf{R}_0$ and $\mathbf{t}_0$ are the camera pose at the first scanline. The function $\mathbf{exp}(.)$:

$so(3) \rightarrow SO(3)$ denotes the exponential map that transforms the angle angle-axis rotation representation to a corresponding rotation matrix. The pose of a given scanline exposed at time τ can be linearly interpolated from Eq. 6.2, which yields the following parametrized transformation matrix:

$$\mathbf{P}(\tau) \quad = \quad \begin{bmatrix}\mathbf{R}(\tau) & -\mathbf{R}(\tau)\mathbf{t}(\tau)\end{bmatrix}. \tag{6.3}$$

Given the continuous time pose parametrization in Eq. 6.3, we can rewrite the global shutter reprojection error in Eq. 6.1 as:

$$\underset{\mathbf{v},\mathbf{\Omega},\mathbf{R}_0,\mathbf{t}_0,\mathbf{X}}{\text{argmin}} \sum_m \sum_n ||\mathbf{x}_{m,n} - \mathbf{K} \cdot \mathbf{D}(\pi(\mathbf{P}_m(\tau) \cdot \mathbf{X}_n))||^2, \tag{6.4}$$

where we optimize over the camera pose at the first scanline of each image ($\mathbf{R}_0$, $\mathbf{t}_0$), velocities ($\mathbf{v}$, $\mathbf{\Omega}$) and 3D structures $\mathbf{X}$. Unfortunately, the optimization based on Eq. 6.4 often breaks for our wide baseline images with weak visual connectivity. This problem is made worst at the end of façades in the scene where feature tracks drop tremendously. To overcome this problem, we introduce an additional smoothness term that enforces pairwise smoothness between neighboring poses:

$$\sum_{(i,j)\in\mathcal{G}} ||\mathbf{P}_i^{-1}(0) \cdot \mathbf{P}_j(0) - \mathbf{M}_{i,j}||^2. \tag{6.5}$$

$\mathbf{P}_i(0)$ is the i^{th} camera extrinsics parameters at the first scanline we optimize over, and represented as a 6 dimensional vector $[\log(\mathbf{R}_0)^\top, \mathbf{t}_0^\top]$ where $\log(.) : SO(3) \rightarrow so(3)$. $(i, j) \in \mathcal{G}$ denotes all pairs of neighboring poses in the pose graph $\mathcal{G}$. Here, $j = i + 1$ denotes consecutive neighboring poses, and $j \neq i + 1$ denotes loop closure neighboring poses mentioned in Section 6.2. For consecutive neighboring poses, i.e. $j = i + 1$, $\mathbf{M}_{i,j}$ is the corresponding relative pose obtained from the GPS/INS reading:

$$\mathbf{M}_{i,j} = \tilde{\mathbf{P}}_i^{-1}(0)\tilde{\mathbf{P}}_j(0), \tag{6.6}$$

where $\tilde{\mathbf{P}}_i(0)$ and $\tilde{\mathbf{P}}_j(0)$ are the absolute poses from the GPS/INS at the first image scanline. It becomes obvious now that Eq. 6.6 penalizes deviation from the GPS/INS poses and preserves smoothness between neighboring poses. For non-consecutive neighboring poses, i.e. $j \neq i+1$, $\mathbf{M}_{i,j}$ is the relative pose computed from the pose estimation during the geometric verification in the loop closure mentioned in Section 6.2. Consequently, our bundle adjustment formulation is also capable of closing large loop closure errors.

Our final objective function then writes as follows:

$$
\begin{aligned}
\underset{\mathbf{v},\boldsymbol{\Omega},\mathbf{R}_0,\mathbf{t}_0,\mathbf{X}}{\operatorname{argmin}} \Bigg\{ \sum_m \sum_n ||\mathbf{x}_{m,n} - \mathbf{K} \cdot \mathbf{D}(\pi(\mathbf{P}_m(\tau) \cdot \mathbf{X}_n))||^2 \\
+\lambda \sum_{(i,j)\in\mathcal{G}} ||\mathbf{P}_i^{-1}(0) \cdot \mathbf{P}_j(0) - \mathbf{M}_{i,j}||^2 \Bigg\},
\end{aligned}
\tag{6.7}
$$

where λ is a weighting factor we estimated empirically and set to 1e6 in all our experiments. We proposed an alternating optimization approach to minimize the cost function in Eq. 6.7. First, we optimize over the parameters $\mathbf{R}_0, \mathbf{t}_0, \mathbf{X}$ while keeping the parameters $\mathbf{v}, \boldsymbol{\Omega}$ fixed. Next, we optimize over $\mathbf{v}, \boldsymbol{\Omega}$ while keeping $\mathbf{R}_0, \mathbf{t}_0, \mathbf{X}$ fixed. We alternate between the two configurations until we reach convergence. In all our experiments, the solution converged after a maximum of two iterations, and in most cases one iteration is enough for convergence.

Fig. 6.2 shows a comparison of the sparse 3D reconstructions (a) without optimization, (b) with global shutter model and (c) with our proposed bundle adjustment cost. It can be seen that the 3D structures reconstructed from GPS/INS readings without bundle adjustment is noisy. Crooked and misaligned facades can also be observed from the model procduced with global shutter camera model. Our proposed cost function for rolling shutter camera produced a clean and sharp 3D sparse model.

6.4 Time Continuous Rolling Shutter Stereo

We adopt and improve upon the plane sweep stereo algorithm proposed in [92] for rolling shutter cameras. The main difference between the plane sweep algorithm for global and rolling shutter cameras is the way that a pixel gets warped from a target image into a reference image for the evaluation of the photo-consistency cost. This can be done easily with homography for the global shutter cameras. Unfortunately, simple homography is not applicable for the rolling shutter cameras since each scanline has a different pose and therefore the plane parameters vary according to the scanlines. [92] incorporates the rolling shutter camera model into the plane sweep algorithm to achieve the warping of a pixel from a target to reference image. This results in the need to solve for the time of exposure τ that corresponds to the time (scanline) a given 3D point is imaged by the sensor. More precisely, the time of exposure τ is obtained by solving the following fix-point function:

$$s \cdot \mathbf{K} \cdot \mathbf{D}(\pi(\mathbf{P}(\tau) \cdot X)) = \tau, \tag{6.8}$$

where $\mathbf{K}$ denotes the camera intrinsics matrix, $\mathbf{D}$ is the lens distortion function, $\mathbf{P}(\tau)$ is the camera extrinsic at time τ and $\mathbf{X}$ is the 3D point and $\pi(.) : \mathbb{P}^3 \rightarrow \mathbb{P}^2$ denotes the projection function. s is the scanline selection operator that selects either the top or bottom row of the left hand side of Eq. 6.8, i.e. we set $s = [1, 0]$ for a shutter that moves horizontally, and $s = [0, 1]$ for a shutter that moves vertically. Note that solving for τ in Eq. 6.8 requires solving for the roots of a high order polynomial, where the order depends on the lens distortion model and motion parameterization. Using the motion parametrization presented in Section 6.3, we get a 9^{th} order polynomial in τ, where τ is solved using the Gauss-Newton minimization. The need to solve a 9^{th} order polynomial for every pixel in the image quickly becomes computationally expensive. Authors of [92] reported a processing time of 27ms per image with the Graphics Processing Unit (GPU). An alternative approach proposed in [92] is to solve τ for a subset of pixels on the target image, and interpolate the full results with the subset of pixels warped onto the reference images. However, this technique comes with the cost of interpolation artefacts. Here, we propose an alternative interpolation scheme where we process every pixel on the target image, but solve τ for a subset of the plane intersections (each plane corresponds to the depth that is searched for pixel correspondence in the plane sweep stereo algorithm) with each back-projected ray from every pixel on the target image. We solve for all the τ values by interpolating between the τ values solved from the subset of plane intersections projected onto the reference image. Fig. 6.3 shows how $\tau(d)$ changes with increasing depth d of the plane in the plane sweep stereo algorithm for a range of 4m to 9m. It should be noted that the closer the intersection point of the ray from the image pixel with the plane from plane sweep gets to the camera, the more rapidly (can be increasing or decreasing depending on the pixel location and camera motion) the time of exposure τ (scanline) changes with increasing velocity of the camera.

Finding a suitable parametrization for the interpolation of τ can be challenging since the curve $\tau(d)$ varies a lot with different pixel location, camera motion and radial distortion. The depth in which we search for pixel correspondences in the plane sweep stereo is bounded by the two planes closest and furthest away from the camera. Exploiting this fact, we can evaluate the τ values for a subset of depths for each pixel and interpolate the missing intermediate τ values. We experimented with different interpolation schemes, such as linear, quadratic, cubic, spline interpolation and piecewise quadratic. The most efficient scheme is found to be the piecewise quadratic interpolation since it has a low memory overhead and is a good approximation of high order polynomials. Other methods are found to be limited to their corresponding polynomial order. Fig. 6.4 shows the error we make when using different interpolation schemes such as linear and quadratic, cubic spline interpolation with 5 knots. Since we only need to evaluate τ coursely in the sweep-space, this gives an additional speed up of $4.3\times$ with a peak pixel error below 1e-2 pixel. While all other interpolation schemes provide an error above 1 pixel, meaning the estimated scanline is off by at least one scanline. As ground truth we use the τ values obtained by solving 6.8 with Gauss-Newton for each pixel.

Since the rolling shutter effect is dependent on the scene depth, the curvature of τ becomes much higher when the plane is closer to the camera, and becomes almost linear when the plane becomes further away from the camera. This motivates to use an adaptive interpolation range in which a quadratic function is fitted in dependence of the relative scene depth. Let c ($\sqrt{3}$ in our experiments) an exponential factor and b be the initial interpolation range. The adaptive depth range d_i used for the piecewise depth interpolation is obtained using $d_i = c^i b$. Given the total number of planes to sweep w, the number of times τ needs to be fully evaluated for each pixel then is: $i = log(w/b)$. In addition we can sparsely evaluate τ in image space and bi-linearly interpolate the values in between. Combining those two interpolation schemes gives us an overall speedup of $6.56\times$ ($7\times$).

6.5 Evaluation

We evaluated our proposed pipeline on both synthetic and on a real dataset. For the synthetic experiments we used the dataset provided by [92].

The real dataset was captured with a car mounted rolling shutter camera rig, consisting of 16 cameras, covering 80% of a sphere. The considered trajectory has a total length of 2.6km and consists of 17k images. Images have a native resolution of 1944×2592 pixels and are sparsely recorded at 4Hz. The shutter scan time for each camera is 72ms. In average the car drives at 17km/h, which results in an average camera motion of 0.34m during image formation, meaning the distance between the first and last scanline is 0.34m apart.

6.5.1 Stereo

For our experiments we use our proposed plane sweep algorithm with a single reference plane normal. The reference plane is obtained by finding the dominant plane using RANSAC [26] in the sparse point cloud, obtained from SfM. A sweep through 3d space is performed within in the distance of $[d_{front}, d_{back}]$ around the reference plane. In our experiment we use $d_{front} = 5m$ and $d_{back} = 3m$. Planes in between are sampled linearly in image space. A pixel transfer between two images is computed by first undistorting a pixel and intersecting the resulting ray with the corresponding plane. The intersection point is then back-projected into the other view for texture lookup. The correlation between two images is computed using NCC, with a window size of 5×5 pixels. To be more robust towards textureless surfaces we make use of a multi-resolution approach. Where we aggregate the correlation cost over multiple pyramid levels (3 levels in our case). Furthermore to handle occlusions we use a multi-view approach. At each depth (plane) we consider the $k = 3$ best views, out of $n = 6$, that provide the highest correlation. Once the cost volume is obtained we regularize it using Semi-Global-Matching [47] using 16 different path directions. To obtain the final depth we fit a quadratic through the depth that provides the lowest cost. As a last step all estimated 3D points undergo a geometric verification. Points which provide a consistent depth within 3 or more views are considered to be valid points. We use a threshold of 0.1m for our consistency check.

Synthetic Data: In Tab. 6.1 we evaluate our algorithm on synthetic data. We show that the proposed method performs comparable to the one of [92], median depth error on both datasets is below 5cm. And the fill rate is slightly lower 0.7% for the castle dataset and 4.8% for the old town dataset. Interpolating the τ value in sweep space using the proposed piecewise quadratic interpolation a speedup of 3.7 is achieved. When also considering the interpolation in image space, similar to FA2 of [92], a speedup of 6.6 (7) is achieved.

		Castle			Old Town		
Method	speed [ms]	MED [m]	MAD [m]	fill rate	MED [m]	MAD [m]	fill rate
FA2 [92]	2.2	1.02	1.02	52.8%	0.26	0.22	58.6%
Bilinear + PQ	4.2	0.05	0.049	75.6%	0.099	0.096	57.1%
PQ	7.4	0.049	0.041	75.6%	0.098	0.096	57.2%
RS [92]	27.7	0.041	0.032	76.3%	0.085	0.077	62.0%

Table 6.1: We compare our method to the synthetic dataset provided by [92]. Piecewise quadratic interpolation (PQ) and Bilinear interpolation with PQ.

LiDAR: Besides the synthetic evaluation we also compare our reconstruction to sparse LiDAR data, which was captured alongside with the image data. We re-project the LiDAR data into the estimated depthmap and evaluated the error of the estimated depthmaps, computed once with the rolling shutter pipeline (rolling shutter bundle adjustment and stereo) and once with a global shutter pipeline (global shutter bundle adjustment and stereo), see Fig. 6.5 and Tab. 6.2.

	Global Shutter MED (m)	Rolling Shutter MED (m)
Fig.6.5(a)	0.42	0.23
Fig.6.5(b)	0.37	0.25

Table 6.2: Depthmap comparison to sparse LiDAR data. Note that there is a constant offset of 0.2m between the LiDAR the rolling shutter depthmap, which is due to miscalibration of the LiDAR to the camera.

It needs to be noted that the LiDAR sensor is not perfectly calibrated to the camera sensors, which leads to some inaccuracy of this experiment. Nevertheless we observe that in general the global shutter reconstruction has a much higher error than its rolling shutter counter part. In general the global shutter approach wrongly estimates the scene depth further away.

6.5.2 Full Pipeline

We evaluate the proposed pipeline on a large scale dataset of 2.6km length. First we use a global window bundle adjustment with smoothness prior to refine the camera poses and 3d structure, as described in 6.3. A robust Huber loss function is used to better handle outliers. Only tracks of size 3 or greater are consider in the bundle adjustment process. After bundle adjustment, we remove points which have a reprojection error greater than 1 pixel. We compare the final bundle adjusted model to the initial one in Fig. 6.2.
In the second stage we run the rolling shutter aware motion stereo algorithm, as mentioned in section 6.4. In all our experiments we adaptively evaluate τ in sweep space at a rate of $d_i = c^i b$. Where d_i denotes the depth interval and $c = 1.5$ and $b = 6$. In addition τ is evaluated sparsely in the image, at every 5^{th} pixel and bi-linearly interpolated in between. We show full dense reconstructions of the San Francisco City Hall and its surrounding in in Fig. 6.6 - 6.14.

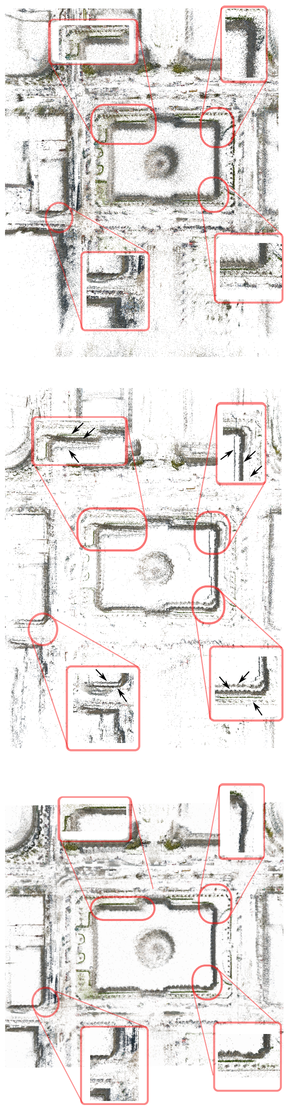


Figure 6.2: Sparse 3D reconstructions. (a) Initial model obtained using GPS/INS pose. In general the sparse point cloud is very noisy. Typical misalignment 0.2 - 0.3m. (b) Global shutter refinement, crooked and misaligned facades (arrows). (c) Rolling shutter refinement (proposed method), sharp and well aligned facades.

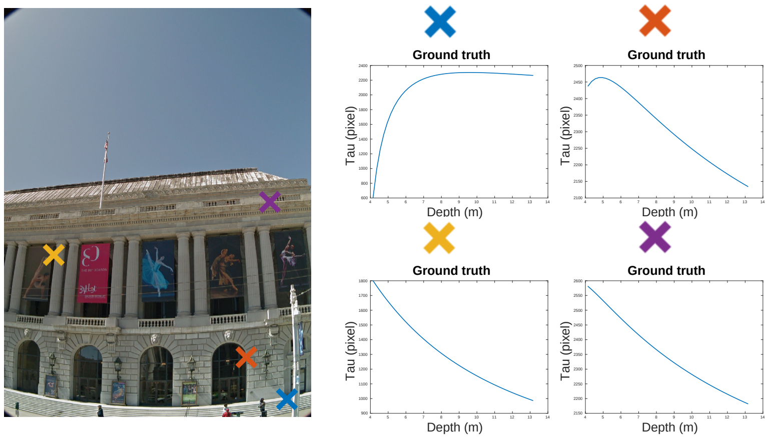


Figure 6.3: Representation of $\tau(d)$ different pixel location over a depth range of 9m, $d \in [4m, 13m]$.

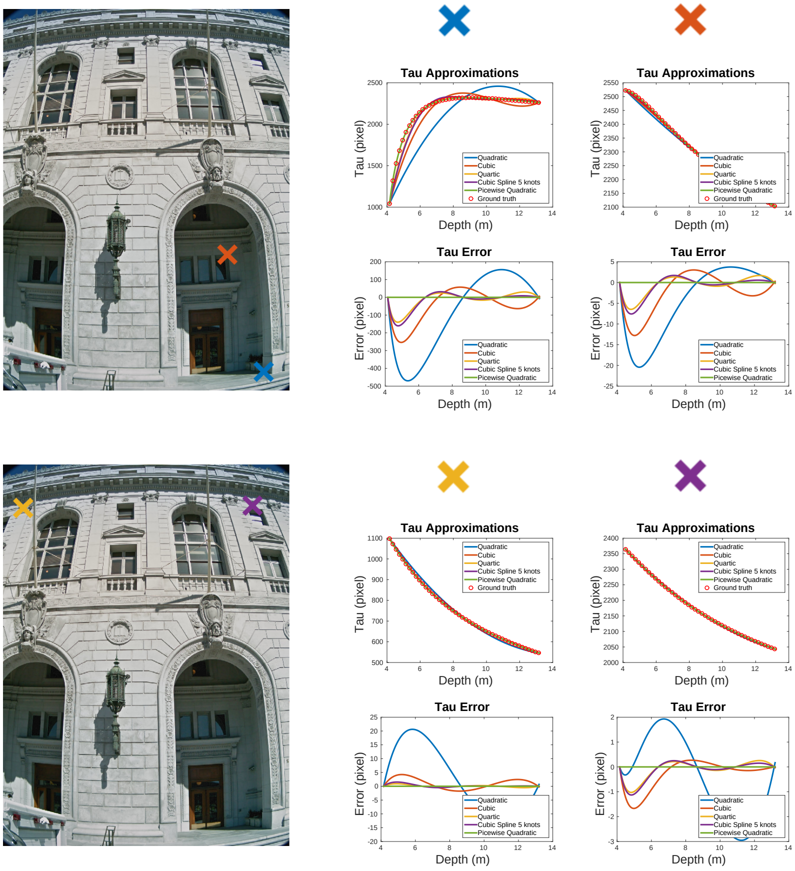
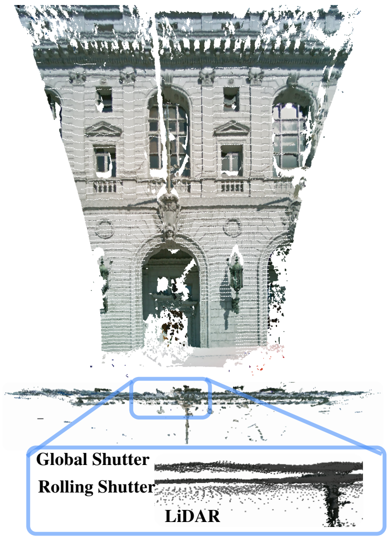
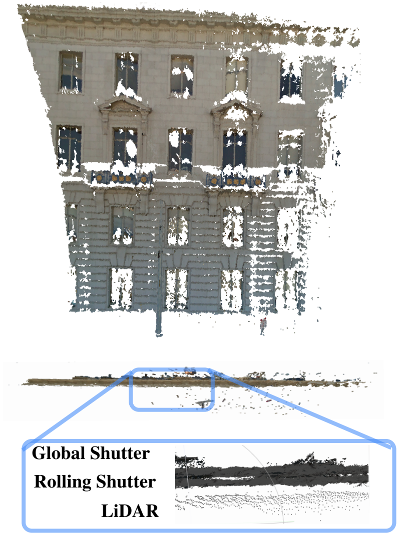


Figure 6.4: Evaluation of τ interpolation error for four different pixels. The first row shows the different approximations of τ in dependence of the depth d, i.e., $\tau(d)$. The second row shows the error to the ground truth. In all examples the four approximations: Quadratic, Cubic, Quartic and Cubic Spline interpolation have at least 1.0 pixel error. While the piecewise quadratic interpolation provides an error below 1e-3 pixel.



(a)



(b)

Figure 6.5: Comparison of rolling shutter and global shutter stereo to LiDAR data. We found that in general global shutter stereo reconstructs objects further away than what they really are. In the above example the depth difference between the two stereo algorithms is 0.2m.

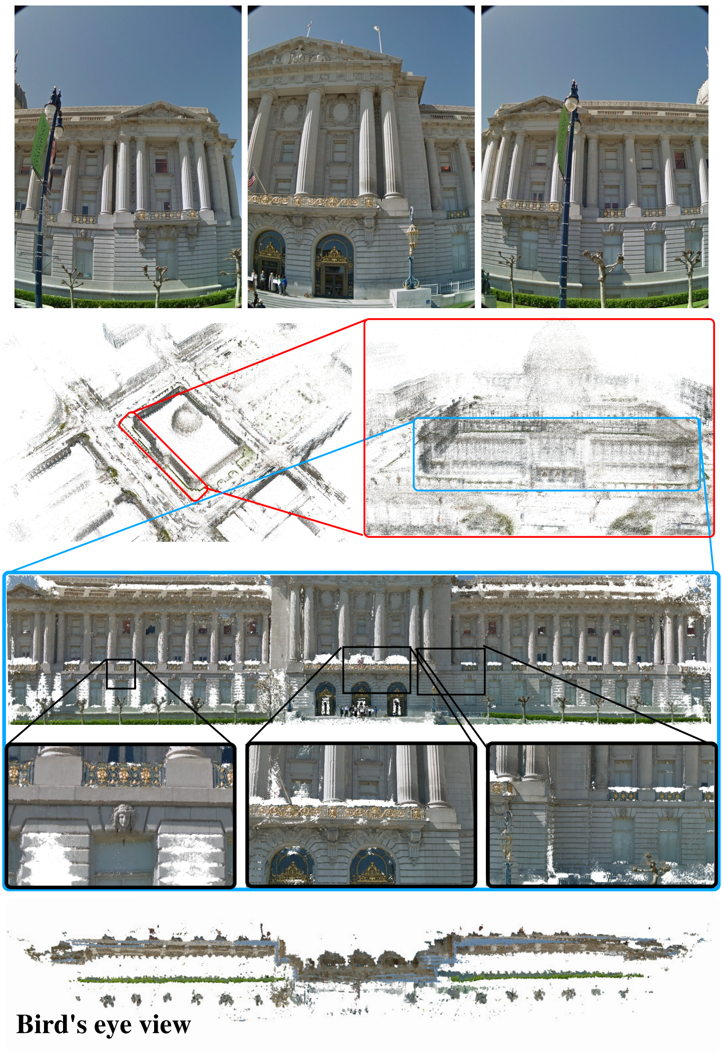


Figure 6.6: Sample reconstructions of the San Francisco city hall and its surroundings.

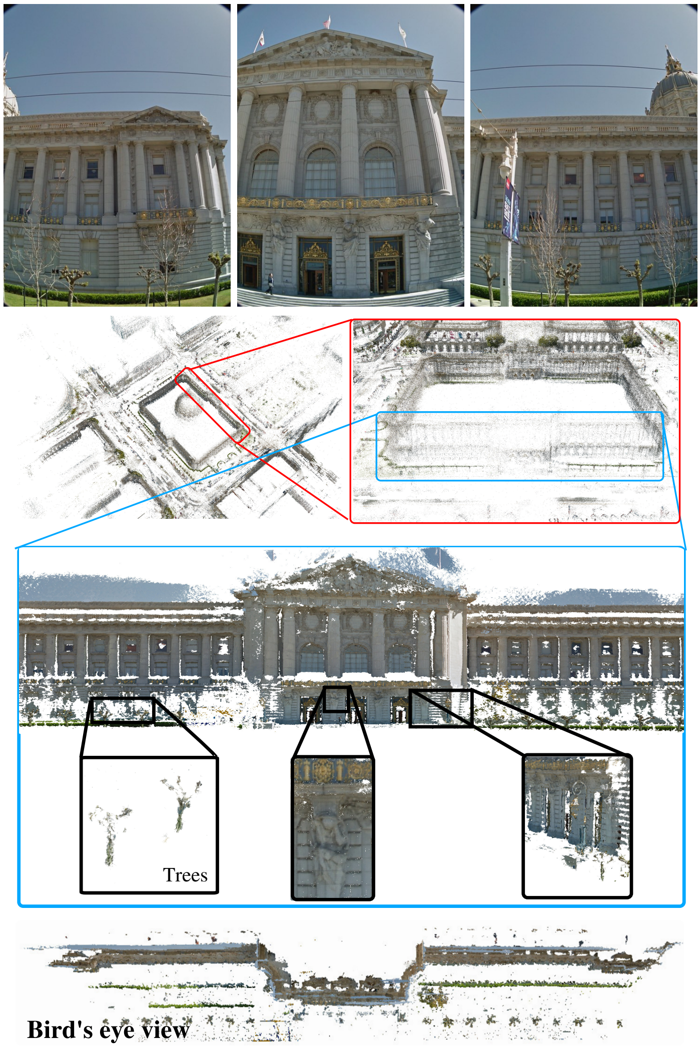


Figure 6.7: Sample reconstructions of the San Francisco city hall and its surroundings.

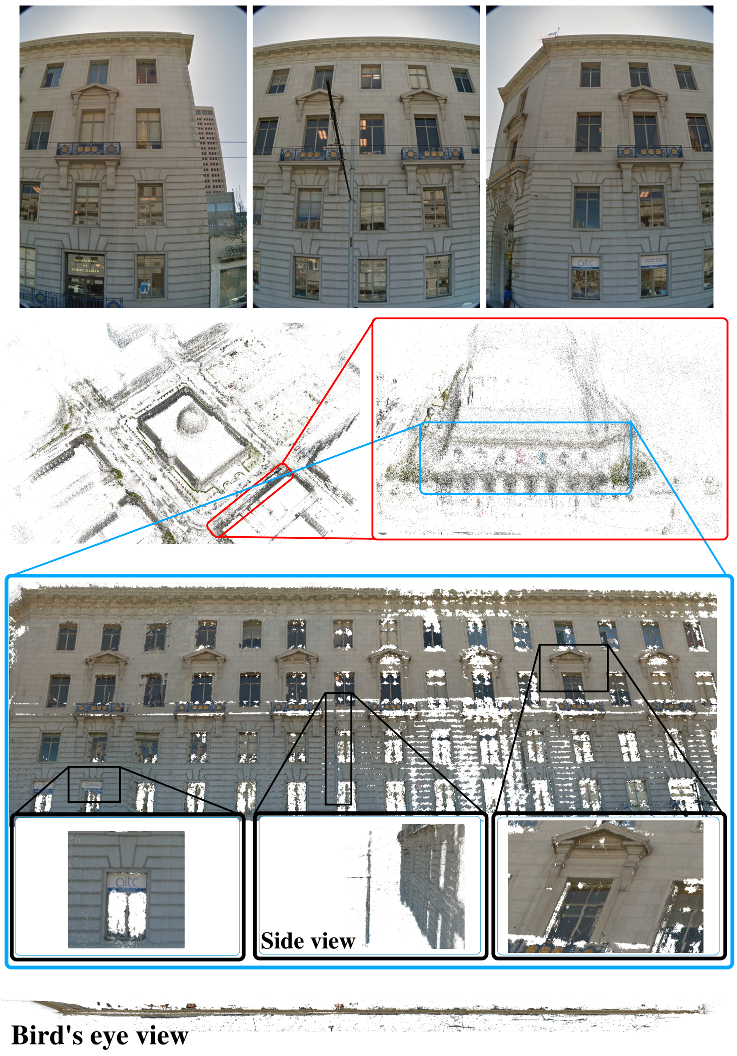


Figure 6.8: Sample reconstructions of the San Francisco city hall and its surroundings.

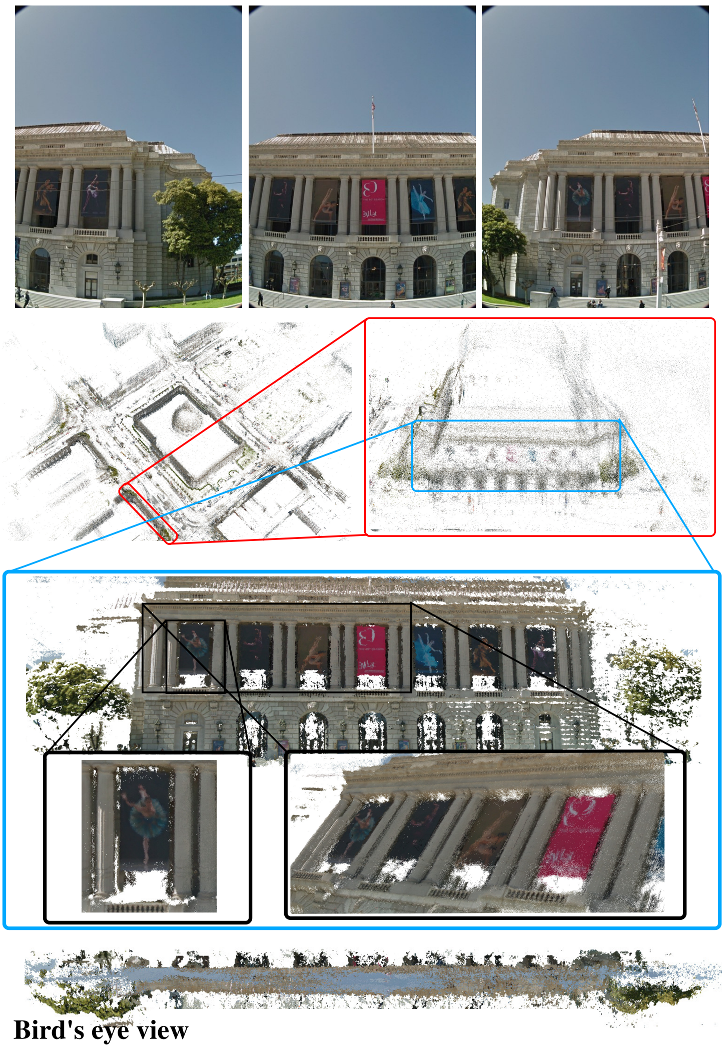


Figure 6.9: Sample reconstructions of the San Francisco city hall and its surroundings.

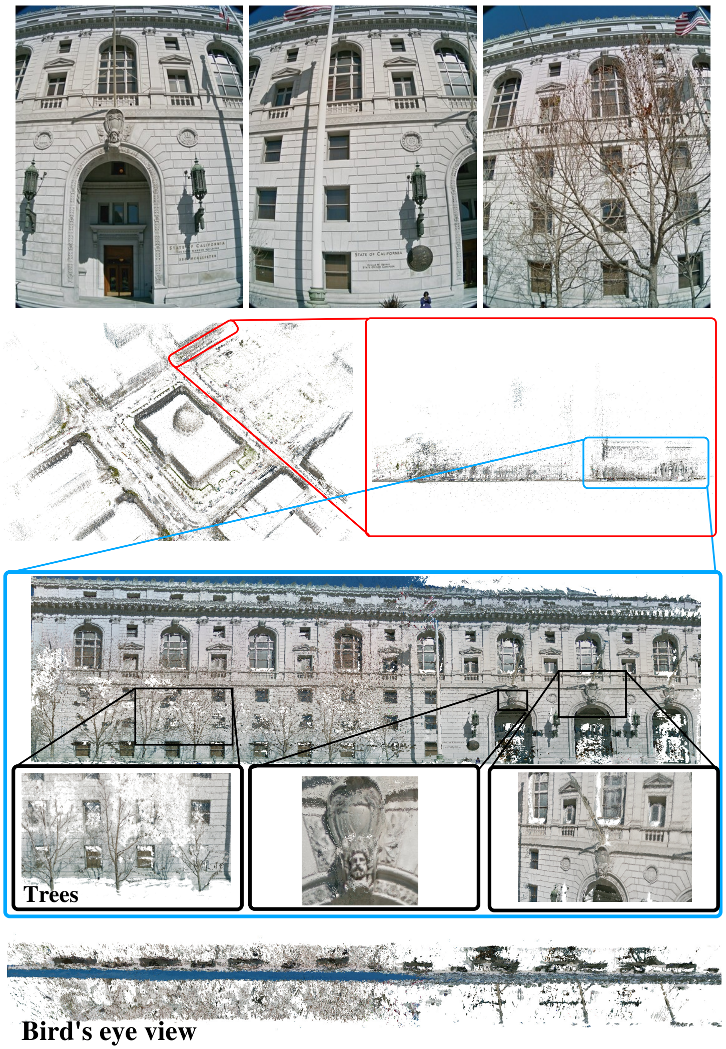


Figure 6.10: Sample reconstructions of the San Francisco city hall and its surroundings.

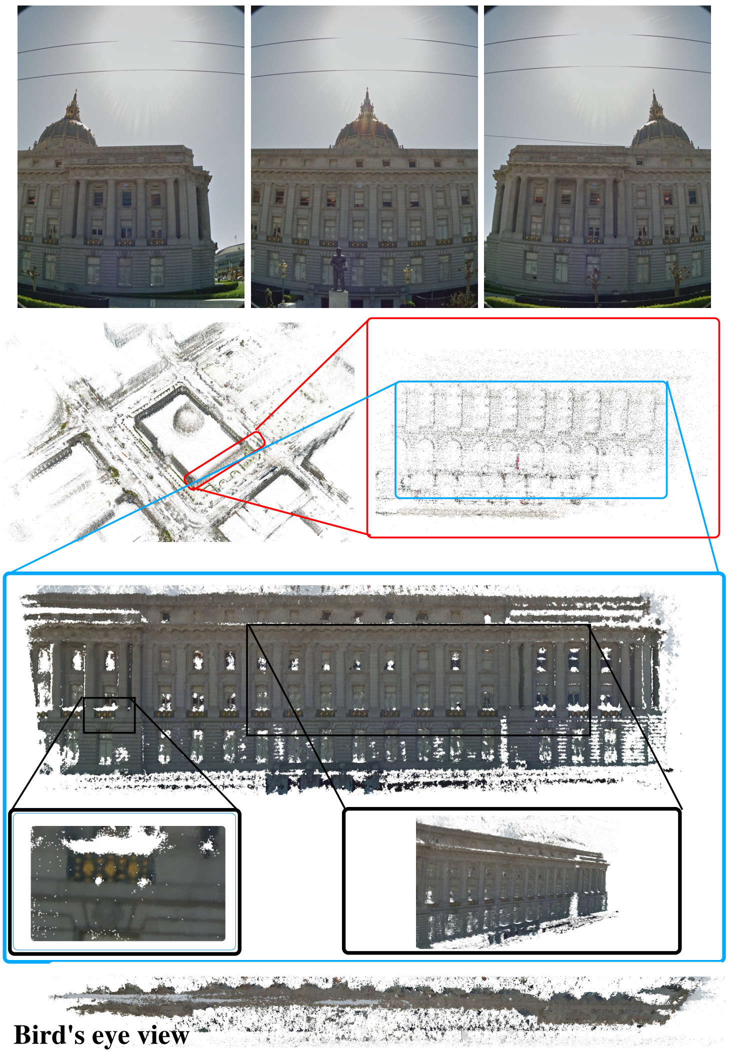


Figure 6.11: Sample reconstructions of the San Francisco city hall and its surroundings.

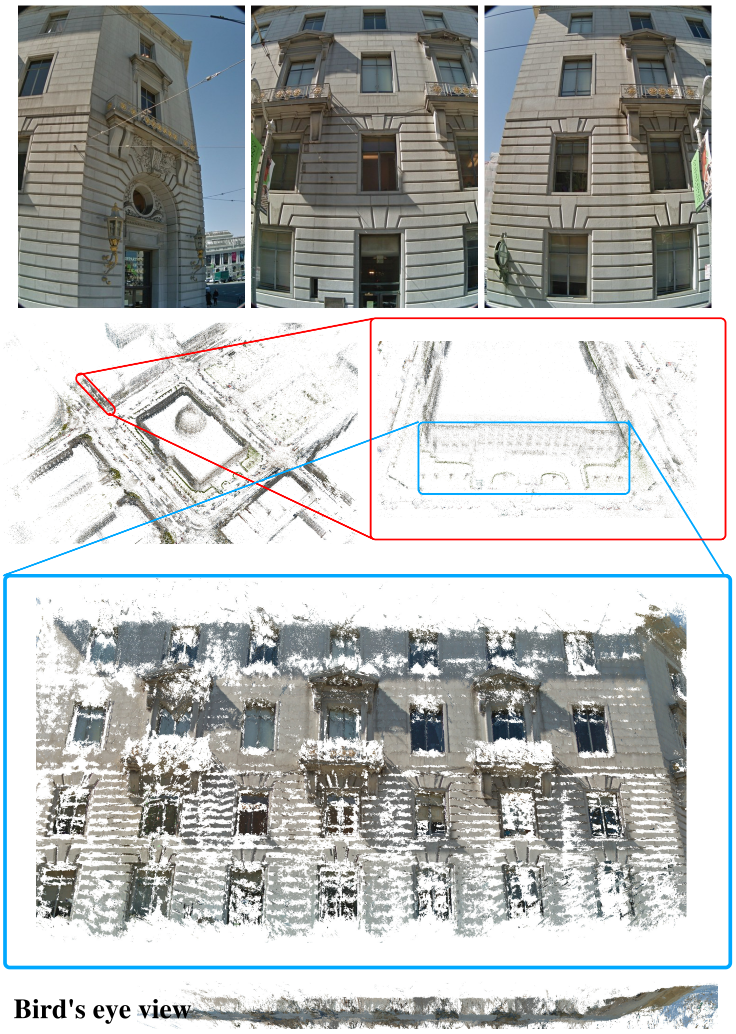


Figure 6.12: Sample reconstructions of the San Francisco city hall and its surroundings.

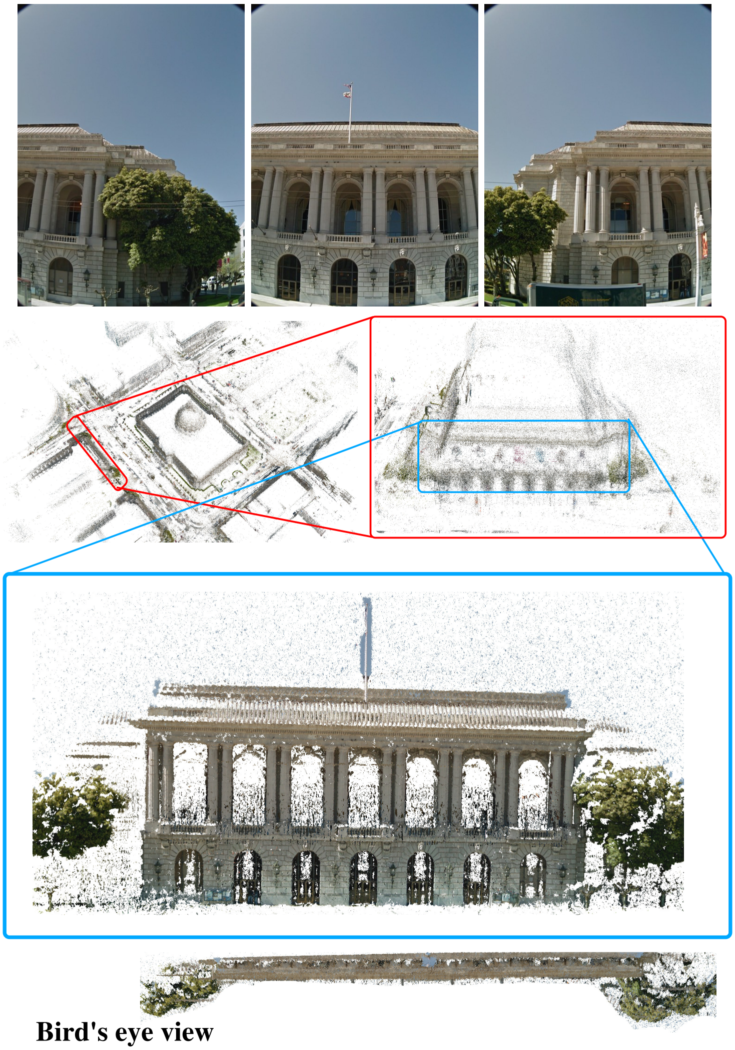


Figure 6.13: Sample reconstructions of the San Francisco city hall and its surroundings.

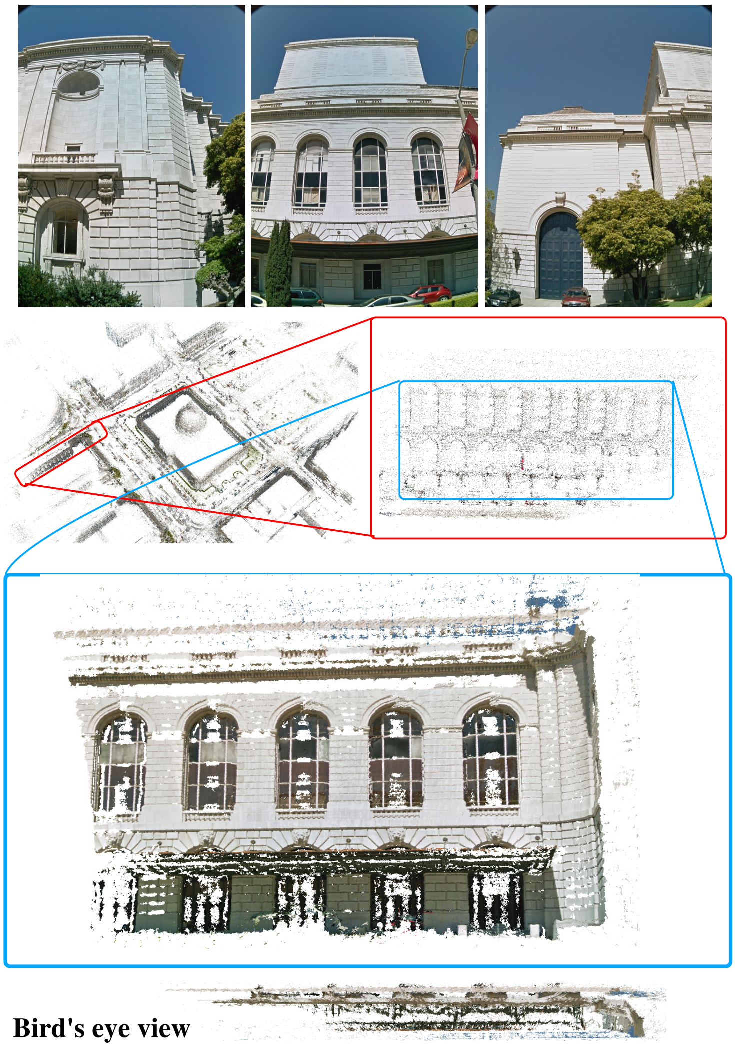


Figure 6.14: Sample reconstructions of the San Francisco city hall and its surroundings.

7 OmniTour: Semi-Automatic Generation of Interactive Virtual Tours from Omnidirectional Video

Contents

The development of Google Streetview [38] really marked a milestone for online map services. From then on it was possible to virtually and interactively walk through cities along roads and experience views as if you were there. The system was deployed on a large scale and with high quality photos. Key features of the system are, that the photos are aligned to road map data and that it is possible to turn when roads intersect. For this, the photos are geo-referenced and aligned with satellite map data using GPS. This allows a user to click on a point in the map and the corresponding view shows up. The map alignment and the detection of intersections are the main challenges of such a system and in Google Streetview these are resolved using GPS annotated photos. In this chapter we propose a system and workflow for Streetview-like virtual tours of indoor environments, e.g. malls, museums, public buildings. Within buildings, geo-referencing with GPS is not possible and thus map alignment and junction detection cannot be done as for the Streetview application. We overcome this limitation by using structure from motion (SfM) and visual place recognition instead of GPS annotated photos. We present a semi-automatic work flow that computes as much as possible automatically and allows manual intervention for a final polishing.

The system works with omni-directional images from a wearable image acquisition setup. As a first step SfM is used to compute an initial camera path. Next a visual place recognition system is used to detect loops and junctions. This information is used to improve the initial camera path by adding them as constraints into an optimization step. Next step is the alignment of the camera path with the floor plan of the building, for which an interactive authoring tool was designed. A user can specify ground control points which align the camera path to the floor plan. This process is interactive, every change is immediately incorporated and the user can see the change instantaneously. After alignment the virtual tour can be experienced with our viewer application.

7.1 Related Work

One of the first virtual tours built, was within the Movie Map project, developed by Andrew Lippman [63] in the $1980s$. The city of Aspen in Colorado was recorded using four 16mm cameras mounted on a car. Where each camera pointed in a different direction such that they captured a 360 degree panorama. The analog video was then digitized to provide an interactive virtual tour through the city.

Now, 30 years later, the process of scanning entire cities or buildings has become much more practical. Image based rendering techniques have increased the interactivity when exploring virtual scenes. In the 1990s Boult *et al.* in [15] developed a campus tour, allowing a user to freely look around while navigating through the campus. The images were taken from a catadioptric camera, where a curved mirror provides a 360 degree panoramic view. While the previous projects focused on outdoor scenes, Taylor's VideoPlus [103] provided an indoor walk through using a sparse image set.

Recently, Uyttendaele *et al.* in [110] proposed a system to create virtual tours using six cameras tightly packed together, to increase image quality. The six camera views are then combined to a single high resolution omnidirectional view using image based rendering techniques. They provide virtual tours through indoor and outdoor scenes. At junctions the viewer can change from one path to an other. Unfortunately, their system does not automatically recognize junctions, instead an author is asked to manually select an intersection range in both paths, then the system performs an exhaustive search to find the optimal transition between both paths, i.e., the transition with minimal visual distance.

Furukawa *et al.* [32] proposed a fully automated reconstruction and visualization system for architectural scenes, based on a sparse set of still images. Their approach is based on extracting very simple 3D geometry by exploiting known properties of the architectural scene. The model is then used as a geometric proxy for view-dependent texture mapping.

Levin *et al.* in [58] proposed a system to automatically align a camera trajectory, obtained from a SfM algorithm, with a rough hand-drawn map describing the recorded path. Similar panoramic views are recognized using a hierarchical correspondence algorithm. In a first step color histograms are matched, then the rotation invariance is verified by computing the 3D rotation between frames. The final frame correspondence is accepted if the epipolar geometry provides enough consistent feature matches. The ego motion is then matched with a hand drawn map and optimized using loopy belief propagation. Their approach is limited to the accuracy of the hand-drawn map, and does not allow user interaction to refine the alignment. Similar to the previous work Lothe *et al.* [66] proposed a transformation model to roughly align 3D point clouds with a course 3D model.

We propose a system which gives a good first estimate of the camera trajectory. We then provide an authoring tool to manually align the camera trajectory with a floor plan. The manual alignment is aided by an optimization algorithm which incorporates both manual and place recognition constraint to solve for an optimal camera trajectory aligned with a floor plan.

7.2 System Overview

An overview of the proposed processing pipeline is given in Fig.7.1. The input to our algorithm is an omnidirectional video stream together with a floor plan. We first convert each frame to six radial undistorted images. After extracting SIFT features we select key-frames based on the motion of the features. Then, we search for already visited places using visual words and filter false frame matches with a hierarchical filtering scheme. The ego motion of the camera is estimated using a SfM algorithm. Finally, the camera trajectory is optimized, as described in section 7.6, which incorporates both place recognition and user supplied constraints.

7.3 Acquisition and Preprocessing

Camera model: The input to our algorithm is an omnidirectional video stream, captured by a Ladybug2[1] camera. The camera consists of six 1024×768 color CCDs. Five of which are positioned horizontally and one is pointing upward. Similar to Tardif *et al.* in [102], we model the Ladybug unit head as a single central projective pinhole camera holding six image planes with different orientations in 3D space. Furthermore, we assume that the principle axis of all six cameras are aligned and intersect in the origin of the Ladybug unit head coordinate system. The projection equation for each of the six cameras is then given by Eq.7.1:

$$\mathbf{x}_i = \mathbf{K}_i\mathbf{R}_i[\mathbf{I}| - \mathbf{C}_i]\mathbf{X}, \qquad i \in \{1, \dots, 6\}, \tag{7.1}$$

where $\mathbf{K}_i$ represents the intrinsic calibration matrix, $\mathbf{R}_i$ and $\mathbf{C}_i$ the rotation and translation of camera i relative to the Ladybug unit head coordinate system. Given the above assumption that $\|\mathbf{C}_i\|$ is negligible, the projection equation rewrites as

$$\mathbf{x}_i = \mathbf{K}_i\mathbf{R}_i[\mathbf{I}|\mathbf{0}]\mathbf{X}, \qquad i \in \{1, \dots, 6\}. \tag{7.2}$$

This assumption results in the change of the focal length for each of the six cameras, making the original light ray flatter. Triangulating 3D points using

[1] http://www.ptgrey.com/products/ladybug2/

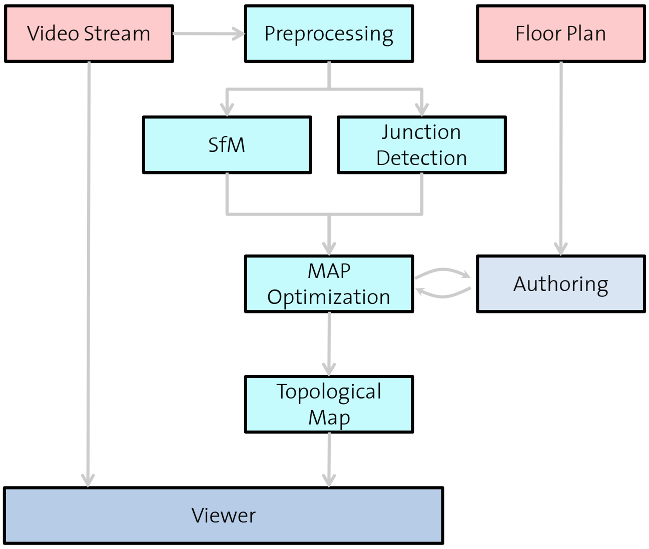


Figure 7.1: Overview of the proposed processing pipeline, for semi-automatic alignment of camera trajectories with a floor plan.

this camera model will result in a wrong depth estimation. However for points far enough this is negligible.

Data acquisition: To acquire our omnidirectional images, we mounted the Ladybug camera onto a backpack. The backpack can be attached to a wheelchair to scan long planar paths, such as corridors and hallways. Stairways and locations not accessible with a wheelchair are recorded by wearing the backpack. The wheelchair setup is in favor, since it gives a smoother camera motion and a more natural height when virtually exploring the building.

Feature extraction: Once a video sequence is captured, each camera stream is converted into a sequence of images. The high lens distortion is corrected using the lens undistortion algorithm provided by the Ladybug SDK. To extract key points we use the SIFT implementation provided by Vedaldi and Fulkerson[2]. We only keep features located inside a bounding box of size 280×315 pixels, centered at the principle point.

Key-frame selection: Key-frames represent a subset of the image sequence, where two neighboring key-frames are separated such that a well conditioned essential matrix can be estimated. A frame is selected as key-frame if more than 10% of its features have moved over a threshold of 20 pixels.

The key frames together with their SIFT features form the input to the SfM and junction detection algorithm outlined in section 7.4 and 7.5.

7.4 SfM using the Ladybug Camera

The input to our SfM algorithm are the key-frames of the omnidirectional video stream which holds a set of discontinuous path sequences.

First, we transform the features of all six CCDs into the Ladybug coordinate system and merge them to a single feature set. Then, we transform the 2D rays into 3D vectors which are normalized to unit length to increase robustness of the algorithm. For each pair of key-frames we compute the camera trajectory by extracting rotation and translation from the essential matrix.

To compute the essential matrix between two frames the 1-point method proposed in [96] is used. It exploits the non-holonomic constraints of our wheelchair setup and the planar motion and thus requires only 1 point correspondence to compute the essential matrix. This makes motion estimation very fast and robust to high numbers of outliers. We omit relative scale estimation as this is usually subject to drift. Instead we rely on the internal loop and junction constraints and the manual input to compute the relative

[2]http://www.vlfeat.org/ vedaldi/code/siftpp.html

scales of the path. The full path is then obtained by concatenating consecutive transformations.

7.5 Loop and Junction Detection

We recognize already visited places, solely based on vision. Our technique requires to be rotation invariant, since the camera might traverse an already visited place pointing into a different direction. Furthermore, finding spatially adjacent frames which are temporally separated, requires a framework which quickly discards a large portion of the input images, to reduce frame correspondence search. We make use of the bag-of-word [79] schemes providing a first guess of a loop-closure, which is verified using a geometric constraint. The visual dictionary used for quantization, was trained beforehand on a dataset containing random images taken from the Internet. The quantized frames are inserted into a database. Then, for each frame the database is queried for similar frames. The potential frame matches obtained from the query are further pruned down using a hierarchical filtering scheme, consisting of a visual word match, SIFT feature match and a final geometric verification. Each stage of the hierarchical filtering scheme can either accept or reject a frame pair. If a frame pair is accepted by one stage it is passed on to the next stage. A frame pair is finally accepted as true match, if they satisfy the epipolar constraint $x_b^\top E x_a = 0$, meaning the two frames share a common view of the same 3D scene. Each true frame match provides one entry in the similarity matrix, encoding the number of feature matches.

The similarity matrix is then post-processed to remove perceptual aliasing matches, characterized by sparse clutter, see Fig. 7.4. We identify sparse clutter by labeling the connected components of the binarized similarity matrix and remove regions which size is below a certain threshold (30 in our experiments).

For each frame we then search for the best match in the similarity matrix. The best match is defined as the image pair with the highest ratio of inliers vs. feature matches. These frame correspondences are then used as constraints in the optimization process.

7.6 Interactive Alignment to Floor Plan

7.6.1 Notation

In the following, camera positions and motions are written as coordinate frame transforms. Camera positions are denoted by the coordinate frame transform from the world origin into the camera coordinate system and written as $\mathbf{E}_i$ for the i-th camera. The motion between camera $\mathbf{E}_i$ and camera $\mathbf{E}_{i+1}$ is denoted as $\mathbf{M}_i$. All coordinate frame transforms consist of a 3×3 rotation matrix $\mathbf{R}$ and a 3×1 translation vector $\mathbf{t}$ and is represented by a 4×4 homogeneous transfromation matrix:

$$\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}. \tag{7.3}$$

In the following derivation, the variables $\mathbf{E}$, $\mathbf{M}$, $\mathbf{V}$ and $\mathbf{N}$ describe a homogenous transformation as formulated in Equation 7.3. In this chapter the transformations and their uncertainties are written in terms of the Lie algebra of $SE(3)$ using the exponential map. This parameterization is extensively discussed in [100] and not repeated here.

7.6.2 Fusing SfM with Ground Control Points

From the SfM algorithm we get the camera path as a sequence of transformations between subsequent cameras. The transformations have 6DOF and are denoted as $\mathbf{M}_0, ..., \mathbf{M}_n$. In the interactive alignment this path needs to be fused with ground control points $\mathbf{V}_0, ..., \mathbf{V}_m$ that are specified by the user. The path needs to be changed so that it goes through the ground control points. The individual camera positions of the path are denoted by $\mathbf{E}_0, ..., \mathbf{E}_n$ which are the results of the fusion. Fig. 7.3(a) shows an illustration of a camera path and the corresponding transformations. Every transformation has an uncertainty specified by a covariance matrix. We are seeking the maximum a posteriori estimate of the transformations $\mathbf{E}_0, ..., \mathbf{E}_n$ which is done by minimizing the following Mahalanobis distance:

$$\begin{aligned} w &= \min_E \left(\sum_i (\mathbf{M}_i - (\mathbf{E}_{i+1} - \mathbf{E}_i))^\top \mathbf{C}_{\mathbf{M}_i}^{-1} (\mathbf{M}_i - (\mathbf{E}_{i+1} - \mathbf{E}_i)) \right. \\ &\quad \left. + \sum_i (\mathbf{V}_i - \mathbf{E}_i)^\top \mathbf{C}_{\mathbf{V}_i}^{-1} (\mathbf{V}_i - \mathbf{E}_i) \right) \qquad (7.4) \\ &= \min_E \Big((\mathbf{M} - \mathbf{HE})^\top \hat{\mathbf{C}}_{\mathbf{M}}^{-1} (\mathbf{M} - \mathbf{HE}) \\ &\quad + (\mathbf{V} - \mathbf{KE})^\top \hat{\mathbf{C}}_{\mathbf{V}}^{-1} (\mathbf{V} - \mathbf{KE}) \Big) \qquad (7.5) \end{aligned}$$

In the first term of Eq. 7.4 $\mathbf{E}_i$ and $\mathbf{E}_{i+1}$ should be computed so that the transformation between the two camera poses matches the transformation $\mathbf{M}_i$ computed from SfM. At the same time the distance between the ground control point transformation $\mathbf{V}_i$ to $\mathbf{E}_i$ needs to be minimized. Eq. 7.4 can be written in matrix form without summation with $\mathbf{H}$ and $\mathbf{K}$ being incidence matrices that specify for each constraint which $\mathbf{E}$, $\mathbf{M}$ and $\mathbf{V}$ transformations are compared to each other. In general, this problem can be solved by non-linear optimization as shown in [3]. A different solution to this problem was proposed by Smith *et al.* in [100]. They proposed a linear time algorithm for the case of a sequential camera path with sparse ground control points. The algorithm works in 3 steps. First, initial estimates for the $\mathbf{E}_i$ are computed by concatenating the $\mathbf{M}_i$ transformations starting from the beginning of the sequence. The covariances are propagated accordingly. Ground control points $\mathbf{V}_i$ are fused into this by combining these 2 measurements if available. In the second step this is done again but starting with the end of the sequence. This results in two measurements for each $\mathbf{E}_i$ which are then combined optimally in a third step. The combination is done by solving Eq. 7.5 for two individual measurements only.

$$\begin{aligned} w &= \min_{\mathbf{E}_{opt}} \Big((\mathbf{E}_j - \mathbf{E}_{opt})^\top \mathbf{C}_{\mathbf{E}_j}^{-1} (\mathbf{E}_j - \mathbf{E}_{opt}) \\ &\quad + (\mathbf{E}_i - \mathbf{E}_{opt})^\top \mathbf{C}_{\mathbf{E}_i}^{-1} (\mathbf{E}_i - \mathbf{E}_{opt}) \Big) \qquad (7.6) \end{aligned}$$

This scheme is also used to combine a transformation $\mathbf{E}_i$ with a ground control point $\mathbf{V}_i$.

$$\begin{aligned} w &= \min_{\mathbf{E}_{opt}} \big((\mathbf{E}_i - \mathbf{E}_{opt})^\top \mathbf{C}_{\mathbf{E}_i}^{-1} (\mathbf{E}_i - \mathbf{E}_{opt}) \\ &\quad + (\mathbf{V}_i - \mathbf{E}_{opt})^\top \mathbf{C}_{\mathbf{V}_i}^{-1} (\mathbf{V}_i - \mathbf{E}_{opt}) \big) \end{aligned} \tag{7.7}$$

For our system we extended the scheme by adding internal loop constraints $\mathbf{N}_{ij}$. These $\mathbf{N}_{ij}$ are transformations between frames i and j that are computed by place recognition. The illustration in Fig. 7.3 depicts a loop constraint $\mathbf{N}_{ij}$. In our fusion these constraints need to be fulfilled too. For this Eq. 7.5 is extended by an additional term.

$$\begin{aligned} w &= \min_{E} \Bigg(\sum_i (\mathbf{M}_i - (\mathbf{E}_{i+1} - \mathbf{E}_i))^\top \mathbf{C}_{\mathbf{M}_i}^{-1} (\mathbf{M}_i - (\mathbf{E}_{i+1} - \mathbf{E}_i)) \\ &\quad + \sum_i (\mathbf{V}_i - \mathbf{E}_i)^\top \mathbf{C}_{\mathbf{V}_i}^{-1} (\mathbf{V}_i - \mathbf{E}_i) \\ &\quad + \sum_{i,j} (\mathbf{N}_{ij} - (\mathbf{E}_j - \mathbf{E}_i))^\top \mathbf{C}_{\mathbf{N}_{ij}}^{-1} (\mathbf{N}_{ij} - (\mathbf{E}_j - \mathbf{E}_i)) \Bigg) \end{aligned} \tag{7.8}$$

$$\begin{aligned} &= \min_{E} \Big((\mathbf{M} - \mathbf{HE})^\top \hat{\mathbf{C}}_{\mathbf{M}}^{-1} (\mathbf{M} - \mathbf{HE}) \\ &\quad + (\mathbf{V} - \mathbf{KE})^\top \hat{\mathbf{C}}_{\mathbf{V}}^{-1} (\mathbf{V} - \mathbf{KE}) \\ &\quad + (\mathbf{N} - \mathbf{LE})^\top \hat{\mathbf{C}}_{\mathbf{N}}^{-1} (\mathbf{N} - \mathbf{LE}) \Big) \end{aligned} \tag{7.9}$$

To solve Eq. 7.9 we extend the original algorithm proposed in [100] as follows. Our data consists of multiple discontinuous path sequences, which are interconnected by place recognition constraints. The sequences are optimized independently and sequentially. Fig. 7.3(b) illustrates the case of optimizing two connected sequences. The illustration contains two paths $\mathbf{p}_1$ and $\mathbf{p}_2$. In a first step $\mathbf{p}_1$ is optimized. When computing the value for $\mathbf{E}_j$ the position of $\mathbf{E}_i$ from the path $\mathbf{p}_2$ is fused with path $\mathbf{p}_1$. Next, path $\mathbf{p}_2$ is optimized and here the transformation $\mathbf{E}_j$ is used to be fused into $\mathbf{E}_i$. This process has to be iterated so that updates in poses and covariances are propagated sufficiently. Place recognition constraints from self intersecting paths are treated in the same way. This extension allows us to use the initial sequential algorithm of [100] for paths with intersections and loops. It does not find the global minimum of Eq. 7.9 but experiments showed that it is an efficient and practicably approach.

7.7 Experiments

To demonstrate our algorithm we recorded a full floor of a building, excluding offices and rooms not accessible to the public. The stream holds over 14k omnidirectional frames resulting in a total of over 84k mega-pixel images. After preprocessing a set of 4k key-frames remain. They form the input to the junction detection and the structure from motion algorithm. Fig. 7.7 shows one omnidirectional frame of our input, after correcting for the lens distortion.

The similarity matrix obtained from the junction detection, represents frame correspondences between frames which lay temporally apart, see Fig. 7.4. Clusters parallel to the diagonal represent paths that were re-traversed in the same direction while clusters orthogonal to the diagonal were traversed in opposite direction. The sparse clutter, around frame $(2040, 2330)$ represents false frame matches which are due to perceptual aliasing. We only show off-diagonal frame matches, since frames temporally close to each other always look similar.

The SfM algorithm is run on the whole stream, which holds multiple discontinuous paths. The beginning of each sub path can then be found by looking at the number feature matches and their corresponding inlier ratio. Neighboring frames, which are spatially apart and therefore represent the ending or beginning of a new path, will have a few feature matches but almost no inliers satisfying the epipolar constraint. Fig. 7.5 shows the total number of feature matches between two consecutive frames and their number of inliers. The sub graph represents the ratio of feature matches and inliers. Single peaks in the ratio graph indicate the beginning of a new path.

We then automatically extract frame correspondences from the similarity matrix to append non continuous paths and introduce loop closure constraints into our optimization. In our experiments 695 constraints were introduced automatically. Fig. 7.6 shows the SfM after optimization together with the final alignment on top of the floor plan.

When combining the SfM trajectory with control points provided by the user, the error uncertainty of the SfM can be guided through the covariance matrix C, Eq. 7.10, where a strong motion in one direction will provide more uncertainty than a small motion, likewise for the rotation, where a big rotation holds more uncertainty than a small one. We therefore linearly adapt the variance in x and y direction depending on the motion. Similarly for the rotation around the z-axis the variance is increased with increasing rotation angle α.

$$\mathbf{C} = \begin{pmatrix} 10^{-6} & 0 & 0 & 0 & 0 & 0 \\ 0 & 10^{-6} & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha \cdot 10^{-3} & 0 & 0 & 0 \\ 0 & 0 & 0 & x \cdot 10^{-2} & 0 & 0 \\ 0 & 0 & 0 & 0 & y \cdot 10^{-2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 10^{-6} \end{pmatrix} \tag{7.10}$$

7.8 Applications

Given the proposed virtual tour building system, the data can be used for different applications. The most prominent application is the virtual exploration of indoor spaces, similar to google streetview. The second application we consider is an indoor navigation system. We discuss both applications in more details in the following two sub sections.

7.8.1 Authoring

The authoring tool, Fig. 7.6 provides a simple and efficient way to align the camera trajectory to a floor plan. The user can adjust both camera positions and rotation. A preview of the selected camera is provided to help the user localize the corresponding position on the floor plan. The core algorithm to support the alignment process is outlined in section 7.6.

We show in our experiments that a user can align a full floor plan within a couple of minutes using a small number of control points. In addition camera poses can be augmented with a label, describing the locations, such as: lecture room name, cafeteria or elevator. The annotation is used to help a user navigate through the building.

7.8.2 Building Explorer

In the work of [98] we developed a viewer tool consists of two windows, a main window showing the environment which is currently explored by the user and a mini-map showing the user's current position and viewing direction on the floor plan. To display the environment, we first render a flat panoramic image from all six camera views which is used to texture a sphere. The mini-map is provided as a navigation help. It displays the current position and viewing direction of the user on top of a floor plan. The mini-map also provides an easy way to quickly change from one place to an other, by clicking the new location of interest.

When exploring the virtual tour, the user can move forward, backward and change direction at places where two paths intersect. Therefore we can not rely on the sequential storage of the video stream, to display the next frame, since neighboring frames, especially at junctions have been recorded at different time instances. Instead we use the camera position to select the next frame. Depending on the user's current viewing direction and position we search the neighborhood for the closest frame located in the user's viewing frustum. The new frame's rotation is adapted such that it matches the current frame's viewing direction.

Synthetic views are generated between neighboring cameras to provide a smooth transition between frames and improve the user experience. The synthetic view is rendered by interpolating the reference images using optical flow. To further enhance the exploration of a virtual tour, the viewer application provides a functionality to guide the user to a certain point of interest. Given a location and a destination the viewer computes the shortest path to the destination and provides navigation directions, such as turns or reach of destination. For this purpose the virtual tour was manually augmented by associating camera poses with locations of interest such as lecture room, cafeteria and student registration office.

7.8.3 Building Navigation

Given the fully aligned and annotated map we can use this information to provide an indoor navigation system. In the work of [18] we implemented such a system. Given an initial location and the destination a shortest path is computed and a turn-by-turn navigation provided to the user.

While the user walks through the building the position is continuously tracked based on a pedestrian dead reckoning. While dead reckoning based on step detection quickly introduces drift in the position estimation. It can be corrected to some extend by exploiting the topology of the building. When a turn is taken the user can be re-located to the closest possible turn. In addition we use the phone's IMU to classify different motions types such as *walking*, *running*, *standing*, *walking upstairs* and *walking downstairs*, which were previously learned from a labeled dataset. Depending on the motion (*walking, running*), different tracking priors can be used or if the *walking upstairs*, *walking downstairs* are recognized the user can be located to the closest possible staircase. Sample images of the system are given in Fig. 7.9.

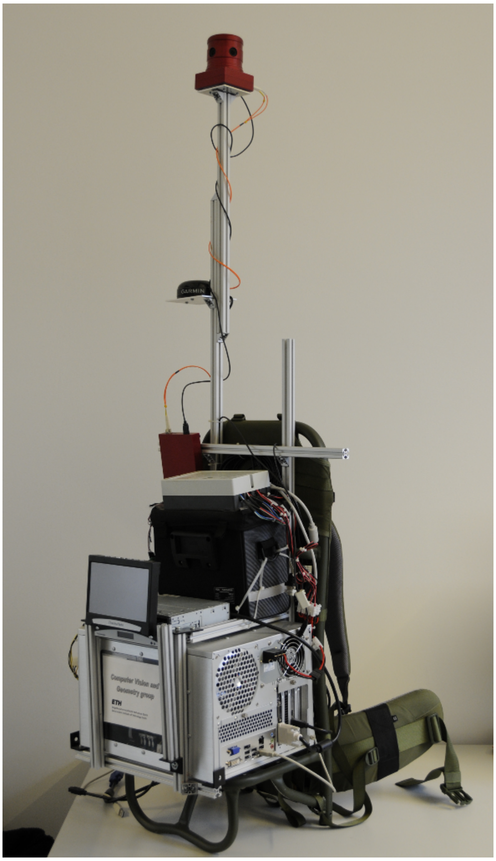


Figure 7.2: Backpack acquisition setup, consisting of a Ladybug2 camera, a small computer to store the captured data and a battery set to power the system.

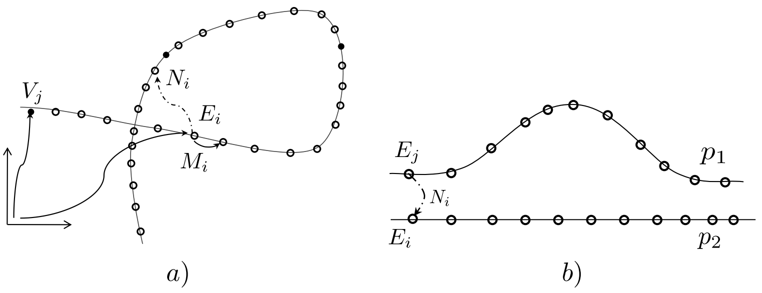


Figure 7.3: V_i denotes the control points set by the user, $\mathbf{E}_i$ denotes the camera translation and rotation, $\mathbf{M}_i$ the motion between neighboring frames and $\mathbf{N}_i$ loop closure or inter path constraints. $a)$ illustrates a self intersecting camera trajectory and $b)$ two interconnected paths.

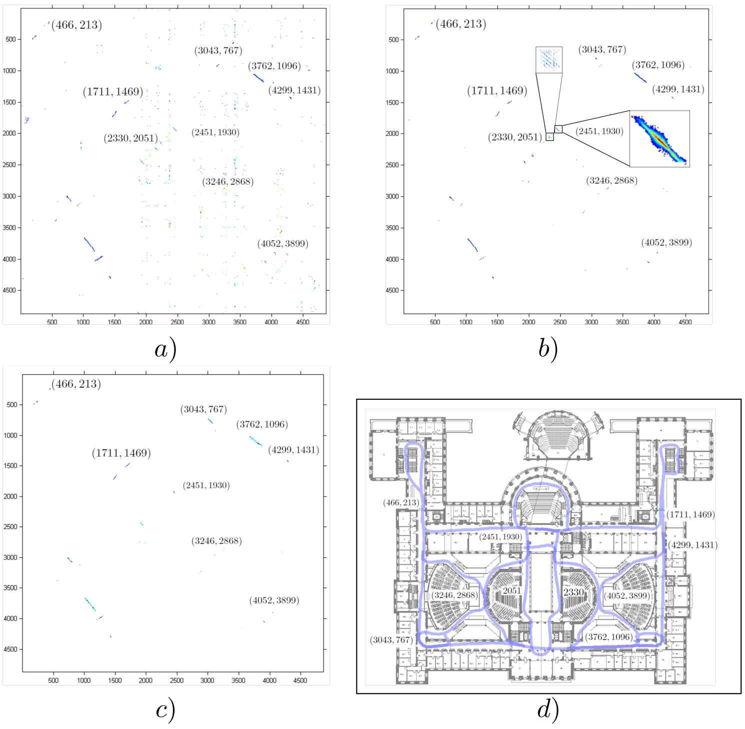


Figure 7.4: The similarity matrix of the processed data. $a)$ similarity matrix after visual word matching, and $b)$ after geometric verification. Note that the sparse clutter around frame $(2040, 2330)$ in image $b)$ represents false frame correspondences due to perceptual aliasing. $c)$ The final similarity matrix after post-processing, i.e., removing sparse clutter. $d)$ the floor plan showing the full aligned camera trajectory with the according frame correspondences.

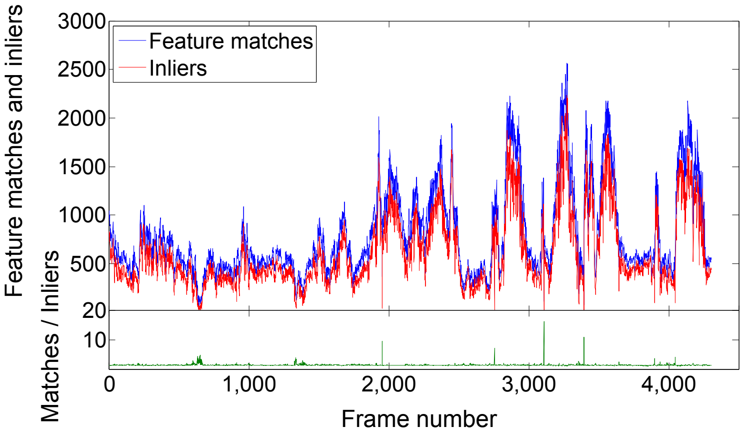


Figure 7.5: The stream holds multiple non continuous path sequences. To find the beginning of a new path, we compute the ratio between the number of feature matches (blue) and the number of inliers (red). The ratio is shown in the subplot in (green). Peak values represent the beginning of a new path sequence, 5 in the processed dataset.

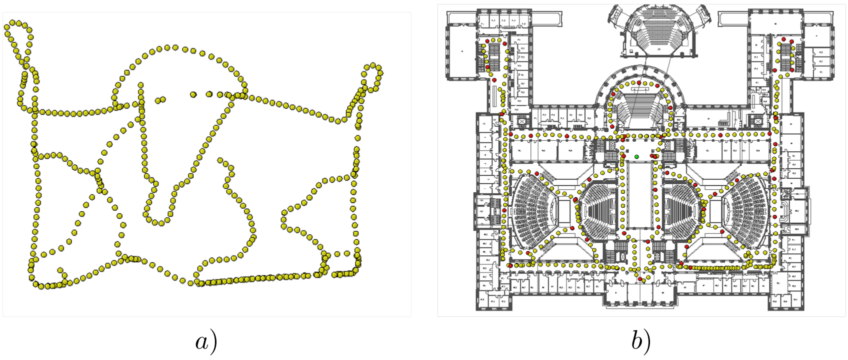


Figure 7.6: In both images only every 10^{th} camera pose is visualized with yellow dots. $a)$ represents the output of the SfM algorithm after applying recognition constraints, obtained from the similarity matrix, without any user interaction. $b)$ represents the final result after aligning the camera trajectory with the floor plan. The point correspondences used to align the camera trajectory with the underlaying floor plan are shown in red.

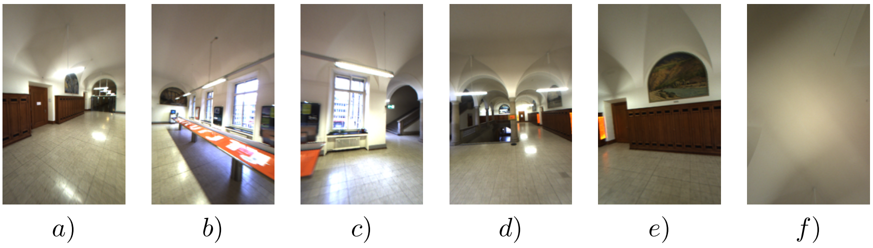


Figure 7.7: One omnidirectional frame. $a)$ - $e)$ represent the five vertical aligned camera views. $f)$ represents the upward pointing camera view. Note that each image has been corrected for lens distortion.

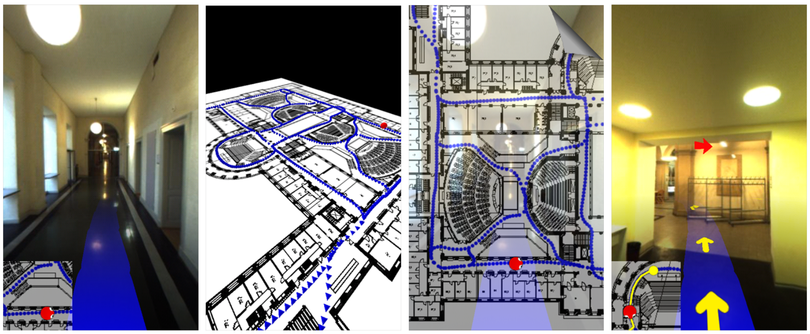


Figure 7.8: Visualization application to interactively explore the virtual tour. A mini-map shows the user's current position and viewing direction on the floor plan. The blue lines show a possible exploration direction. Right, yellow arrows indicate the shortest path to a location of interest. Red arrows indicate turns at bifurcations.

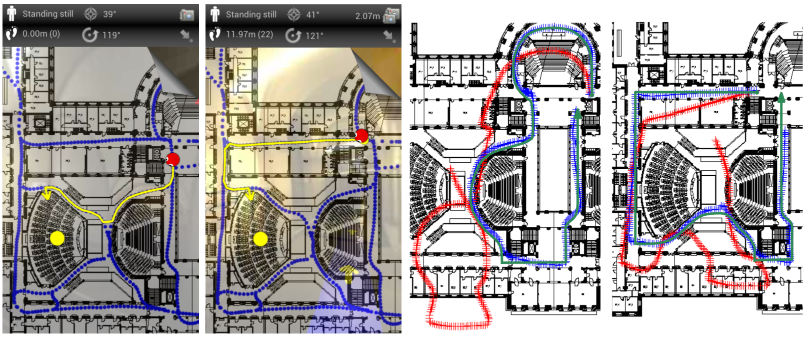


Figure 7.9: Navigation system. Left, shortest path augmentation, from source to destination. Right, alignment correction using the building topology. Red, pedestrian dead reckoning. Blue pedestrian dead reckoning with incorporated building topology. Green, travelled path.

8 Visual Localization using Global Visual Features and Vanishing Points

Contents

Recent approaches to visual robot localization using local image features and visual words proved to work very well [9, 23, 30, 97]. An underlying assumption for these methods however is, that one already collected images for all possible locations in a database. A scenario, where a database was created using images of one floor of a building and having the robot localize itself on a different floor of the building would be beyond the capabilities of these methods. This is exactly the scenario that was created for the ImageCLEF 2010 RobotVision competition [86]. The goal was to train the robot with locations (e.g. office, kitchen, printer room) from one floor, so that it can identify the corresponding locations on the other floor, where the locations differ in details like different chairs, different desks, different posters, different curtains, etc. In this chapter we describe an approach that is targeted towards resolving this scenario. The approach works by using a global image descriptor that captures the large scale features of the location, but not the fine details. This would allow to match up two locations that share the similar overall structure but differ on the fine details. Fig. 8.1 illustrates this concept. The two images show two meeting rooms from the different floors. The table, chairs and pictures on the wall are different but the overall structure is similar. There is a table in the center of the room, which creates a strong horizontal edge feature. The outline of the room walls itself creates also strong edge features converging in a similar manner. These are the features that we would like to capture. To achieve this our approach uses GIST [80] as a global visual descriptor. In addition to visual similarity we propose a subsequent geometric verification check. For geometric verification we compare the vanishing points of matching images, which are computed from line features in the images. This geometric check ensures, that images are matched up only, if they are taken in the same geometric setting (e.g. a similar sized room) and from the same viewpoint.

In the experiments using the dataset of the ImageCLEF 2010 RobotVision competition [86] we demonstrate that using GIST it is possible to capture these larger scale similarities and that it is possible to match up the locations like the one depicted in Fig. 8.1. We also show that the vanishing points are useful for geometric verification and improve the localization results. Finally we report the scores achieved in the ImageCLEF 2010 RobotVision competition.

8.1 Related Work

The GIST descriptor used in our approach was first introduced by Oliva *et al.* in [80]. It was used in [105] for place and object recognition. They showed that it is possible to distinguish between different places or rather scenes using the GIST descriptor. In particular they presented classification results on the following scenes: building, street, tree, sky, car, streetlight, person. In our current work we show that it is possible to use GIST for place recognition in typical indoor environments. In addition we added a geometric verification step targeted to indoor environments. GIST was also used in [75] for place recognition using panoramic images. There the GIST descriptor was adapted to the properties of panoramic images.

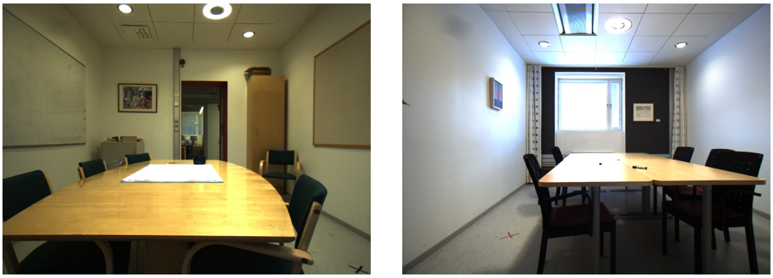


Figure 8.1: Two images from the location class 'Meetingroom' on different floors. To identify these two images as matching an image descriptor needs to identify the large scale similarities (room configuration, table position) despite the obvious differences on the small scale.

8.2 GIST Descriptor and Vanishing Points

Before presenting our pipeline for semantic labeling of space, we first discuss the GIST descriptor which was introduced by Oliva *et al.* in [80]. The GIST descriptor represents scenes from the encoding of the global configuration, ignoring most of the details and object information present in the scene. We then further discuss the concept of vanishing points, which are projections of points laying at infinity. They provide information on the relative camera orientation with respect to the scene and are used as a geometric verification after image retrieval using the GIST descriptor.

8.2.1 GIST Descriptor

The GIST descriptor was proposed by Oliva *et al.* in [80] for scene categorization without the need for segmentation and processing of objects. The structure of the scene is estimated using a few perceptual dimensions such as naturalness, openness, roughness, expansion, ruggedness which describe the spatial properties of the scene. The dimensions are reliably estimated using spectral and coarsely localized information, where membership in semantic categories such as streets, highways, etc. are projected close together in a multidimensional space. The low dimensional representation of a scene is represented by a 960 dimensional descriptor, which allows quick retrieval of similar images from a large database. In the following, image search consists in finding the set of images with the smallest L2 distance.

8.2.2 Vanishing Points

The premise to find vanishing points are man-made environments containing parallel straight lines. When looking at the perspective projection of three dimensional parallel lines, they intersect in one common point in the image plane, the so called vanishing point (VP) [44]. Vanishing points therefor represent the projections of 3D points laying at infinity, since parallel lines intersect at infinity.

To estimate the vanishing points, we first detect edges using canny edge detection and extract long straight lines from the edge segments. The straight lines are used as input for our RANSAC (random sample consensus) algorithm, which estimates multiple vanishing points in a single image. The algorithm first randomly selects two lines and computes their intersection point P. If at least 20 of the lines passes through the intersection point P, the point is re-estimated using a non-linear optimization, where all supporting lines are included in the optimization process. The supporting lines are then removed from the input set and the procedure is repeated until either no further lines are available or no further vanishing point is found.

Fig. 8.2 shows line sets supporting different VPs. Each color represents the support of a different VP.

8.3 Place Recognition

The proposed pipeline for semantic labeling of space is illustrated in Fig. 8.3. The method classifies an image into one of the following three categories, which is either a label learned from a training dataset, the ”Unknown” label or in some cases the algorithm would make no decision.

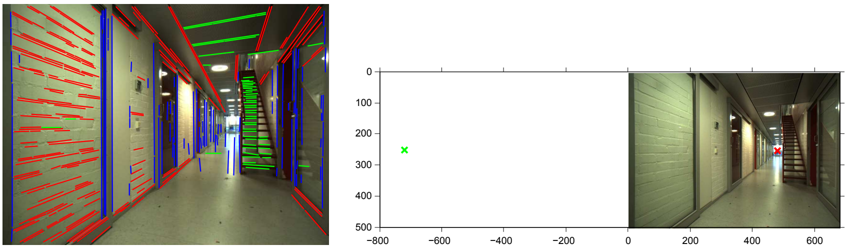


Figure 8.2: Left, lines are classified to the according VPs. On the right the VPs are shown, except the one located far from the image center (infinity).

In a first step a database of GIST descriptors is build from the training dataset. Our database consists of 4780 images and is represented by a kd-tree for fast k-nearest neighbors search, we chose k to be 10 in our experiments. In a first step we query the database with the query image q, for its 10 nearest neighbors stored in the result set r. Images in the result set r with a L2 distance to the query image q, greater than a given threshold (0.6 in our experiments) are removed from r. If r is empty, the image q is labeled as ”Unknown”. Otherwise the set r is further matched to a set of ambiguous images, which were previously learned from the training dataset, see Fig. 8.4. If the set of ambiguous images in the set r is greater than the set of non-ambiguous images, the algorithm refrains a decision on the image q, due to lack of confidence. Otherwise, a geometric verification is applied to the remaining set of non-ambiguous images. The geometric verification compares the angular distance of vanishing points between the query image and the non-ambiguous images. Images with a large angular distance (0.34 in our experiments) are removed from the set r. Finally, the query image is assigned the label of the image with the smallest angular distance or is assigned the label ”Unknown” if the set r is empty.

To find vanishing point matches between two images, we first normalize the VP vector to unit length, such that the VP lays on the surface of a Gaussian

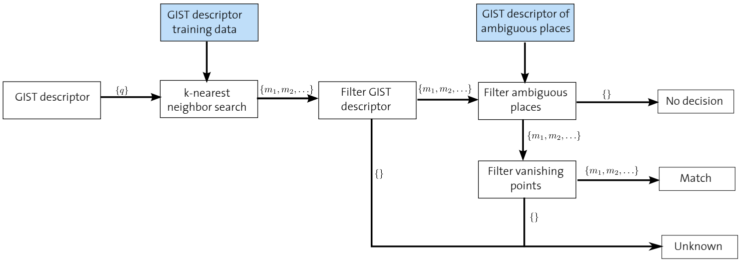


Figure 8.3: Overview of the proposed pipeline.

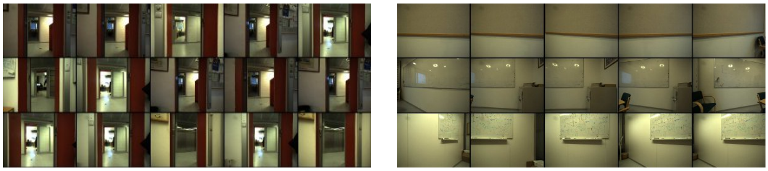


Figure 8.4: Ambiguous places are represented by similar GIST descriptors with different image labels. We have trained two classes of ambiguous images, door frames (left image) and walls, whiteboards (right image).

sphere. Then, for each VP in one image we do an exhaustive search for the closest VP in the other image i.e., the VP with smallest angular distance. We assume that two similar scenes match, if their appearance is similar i.e., similar GIST descriptor and similar vanishing points, meaning the camera has a similar point of view of the 3D scene being observed.

8.4 Evaluation

Our algorithm was evaluated at the 3rd edition of the Robot Vision challenge, held in conjunction with IROS 2010. The challenge addresses the problem

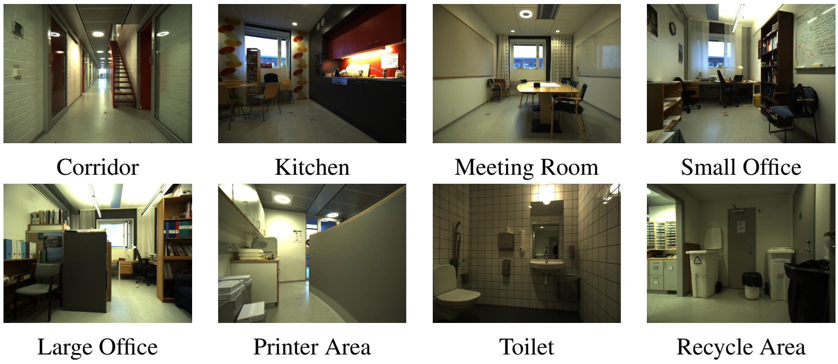


Figure 8.5: Sample images from the training dataset.

of classifying rooms and functional areas based on a pair of stereo images. Three image sets were provided, one training set for learning, one validation set for the participants to validate their algorithm and one testing set used for the competition. All three sets were captured in the same building, but on different floors. All three floors have a set of common rooms, such as *Offices, Toilet, Printer Area, Corridor, etc.* and rooms which are only present in one of the dataset such as *Kitchen, Lab, Elevator, etc.*. Sample images of the training sets are provided in Fig. 8.5.

Task 1 of the competition asked to build a system which can answer the question ”Where am I?”, given one pair of stereo images. The answer can either be a previously learned label, the ”Unknown” label if the system is presented with a new location not contained in the training set or it can refrain a decision by leaving the image unclassified. The performance of the algorithm was evaluated using the following scoring scheme:

- +1.0 point for each correctly classified image.
- -1.0 point for each misclassified image.
- 0.0 point for each image that was not classified.

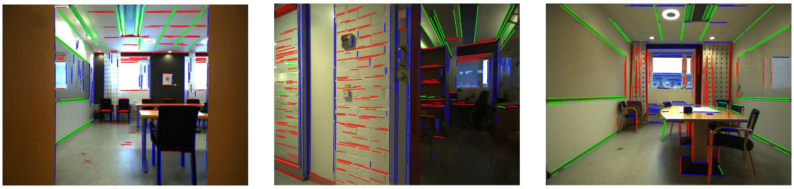
- +2.0 points for a correct detection of unknown category.
- -2.0 points for an incorrect detection unknown category.

Our system ranked first, with 677 points in the 3rd edition of the Robot Vision challenge. The winning run used the following configuration: a search window size of 10 images, a minimum GIST distance threshold of 0.6, and a minimum mean angular distance threshold of 0.34. Door frames, walls and whiteboards were learned and added to the ambiguous location set as well as the following four rooms *Kitchen, Small Office* and *Large Office*.

Bellow we further discuss the benefit of the geometric verification. The evaluation is based on the validation set, which contains 2069 images, where 14.4% of the image labels are unknown to the training set. Without geometric verification an image match is obtained by searching the training set for the image with the smallest L2 GIST distance. Using the geometric verification an image match is obtained by choosing the image with the smallest mean angular distance between the query image and the images obtained from the k-nearest neighbors, with $k = 30$. Table 8.1 lists the recognition rate of each category known to the training set. Overall, the geometric verification performed slightly superior (recognition rate of 43.15%) to the pure GIST based method (recognition rate of 42.03%). The *Meeting Room* category achieved an improvement of over 8%.

For the label *Corridor* and *Large Office* the pure GIST method performs better. The reason our method provides a lower performance on the *Corridor* category is that many images are misclassified at transitional places, where the robot moves from the corridor into a room. Fig. 8.6 illustrates such a misclassification. In the *Large Office* category the misclassified images are mainly classified as *Kitchen* or as *Small Office*. Fig. 8.7 illustrated a misclassification based on the GIST method, which is corrected by the geometric verification.

Our unoptimized Matlab implementation takes 51.21 seconds on a 2.66GHz Core2 Quad CPU, to classify 2069 images using precomputed GIST descriptors and precomputed vanishing points. Extracting GIST descriptors takes in average 1.91 seconds on a 487×487 pixel image. We make use of the



Corridor (a) | Corridor (b) | Meeting Room (c)

Figure 8.6: False image match due to ambiguous labeling of the training set. (a) shows the original query image. (b) shows the image with the smallest GIST distance 0.41 and a mean angular distance of 0.28. (c) shows the image with the smallest angular distance, GIST distance 0.46 and a mean angular distance of 0.01.

freely available Matlab code provided by *Antonio Torralba*[1]. We use our own Matlab implementation to extract vanishing points. In average it takes 0.65 seconds to extract the vanishing points of one image.

[1] http://people.csail.mit.edu/torralba/code/spatialenvelope/

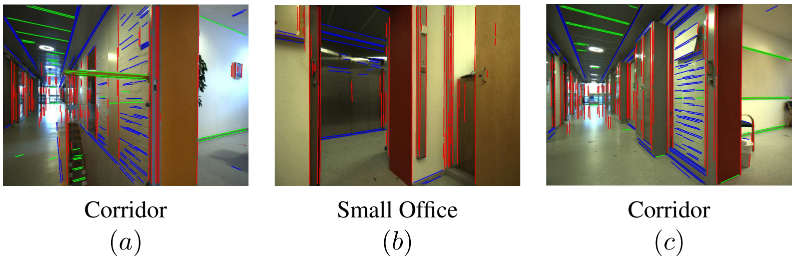


Figure 8.7: Correct image match after geometric verification. (a) shows the original query image. (b) shows the image with the smallest GIST distance, 0.39 and a mean angular distance of 0.37. (c) shows the image with the smallest angular distance, GIST distance 0.47 and a mean angular distance of 0.01.

	Without Geometric Verification	With Geometric Verification
Corridor	74.31%	72.85%
Kitchen	0.00%	0.00%
Large Office	23.41%	19.93%
Meeting Room	44.44%	52.77%
Printer Area	40.21%	40.21%
Recycle Area	47.94%	52.05%
Small Office	14.50%	20.54%
Toilet	84.61%	91.20%

Table 8.1: Recognition rate obtained from the validation dataset. Note that the validation set does not hold the label *Kitchen*, therefor the recognition rate for that label is 0.00%. See text for more details.

9 Conclusion

In this thesis we have proposed several algorithms to estimate camera poses from a sequence of images and a known gravity direction, compute sparse and dense 3D models from rolling shutter imagery and ways to create virtual tours of indoor environments.

The vast majority of today's smart phones are equipped with diverse number of sensors, such as GPS, barometer, illumination sensors, image (CMOS) sensors, IMU (gyroscopes and accelerometers) to name a few. The IMU sensor comes in specially handy in the camera pose estimation process since it provides two orientation angles of the phone relative to the gravity direction. This information can be used when estimating the relative or absolute pose of the camera, reducing the algorithmic complexity. While traditional algorithms use 5 point correspondences to estimate the relative pose, our methods use only 3 point correspondences in the general case and 2 point correspondences if a dominant scene plane is known. Reducing the minimal number of required point correspondences, reduces the total number of RANSAC iterations which leads to a great speedup of the pose estimation algorithm.
The achieved results (chapter 3) on real world experiments demonstrate that the assumptions about known plane normals (ground plane and vertical walls) holds in typical usage scenarios. Our algorithms have successfully been used for indoor robot navigation as well as for 3D reconstruction from aerial images captured by an UAV.

The majority of image sensors used today, especially in mobile phones, make use of the CMOS technology. While this technology has many advantages, such as being cheap to produce, being more light sensitive than it's CCD competitor and in general suffers less from blooming artefacts, it has the peculiarity of exposing each scanline sequentially, so called ”rolling shutter”.

This sequential exposure introduces a temporal delay in the image formation process, which becomes specially apparent when either the camera is moving relative to the scene or when recording an agile dynamic scene. In both cases undesired distortion artefacts are present in the captured images.

Using standard pose estimation algorithms on such imagery leads to wrong and inconsistent camera poses, since only one pose is estimated for the entire image. Since each scanline is exposed at a different time, the camera pose differs with each scanline. We show that the time-delay present in the image formation process, can be exploited to estimate the absolute camera pose together with the camera velocity.

The proposed algorithm requires a minimal set of 5 2D-3D point correspondences. The solution is derived based on Gröbner Basis and can give up to 8 real solutions. Finally, we relax the linear velocity assumption and do a non-linear refinement on the full 12 degree of freedom (translation and angular velocity, and the camera pose) of the rolling shutter camera.

In chapter 5 we analysed the setting of camera *motion* induced rolling shutter effects and have shown that already for very moderate speeds and resolutions, rolling shutter distortions are significant and can break standard stereo algorithms. In particular, although global shutter algorithms *seem* to work out well (resulting in a dense, smooth depth map) the results are actually not correct. We then generalized the homography transfer across a plane known for global shutter cameras to the setting of rolling shutter, considering also lens distortion that intertwines with the rolling shutter. Based on this building block, a plane sweep approach has been implemented that was shown to produce correct results on real and synthetic data. We have furthermore analysed two approximations that provide a significant speedup at the cost of reduced accuracy and analysed the structure of the residual error. This allows to decide for speed or precision in case the rule of thumb presented indicates a rolling shutter model should be used for the setting at hand.

Images that were captured with car mounted cameras usually also come with GPS/INS poses, which were captured alongside. While those pose give already a very good estimate of the camera trajectory they are not precise enough for accurate 3D reconstruction.

Chapter 6 proposes a rolling shutter bundle adjustment, which uses as input GPS/INS poses and optimizes over the full 3D structure, camera poses and velocities. Due to the sparse visual connectivity and sometimes even missing visual information (for instance when the camera images only sky at the end of a block) reconstruction can become crooked or can even break apart. To over come this burden we proposed to include an additional smoothness term in the optimization process, to obtain a complete and consistent model. We evaluate our propose pipeline on a camera trajectory of 2.6km length and show quantitative results of the sparse and dense reconstruction.

While the previous work focuses mainly on outdoor scenes, we also explore the reconstruction of indoor environments. Chapter 7 proposes a full system to record and create topological maps from an omnidirectional video stream. While the reconstructed camera trajectory suffers from drift and lacks the overall scale, it cannot directly be overlaid with the floor-plan of the building. To align reconstructed camera trajectory with the floor-plan, we proposed a user guided optimization scheme. In general the system is not limited to indoor scenes but could also be used in outdoor environments, where control points could be provided from GPS. We have shown that the constructed tours can be used for indoor navigation, providing the user with a turn-to-turn navigation system.

Indoor scenes often consist of texture less regions, such as uniformly colored walls, floors or ceilings and reflecting surfaces (windows), which make keypoint based matching techniques brittle. In addition indoor scenes often consist of movable furniture, which breaks any localization algorithm which relies on the scene geometry. In chapter 8 we address this problem using global visual features such as GIST. To verify the geometry, we use vanishing points, to ensure a consistent orientation between the query image and the database image. We show that this type of geometric verification can indeed improve the recognition rate when used in conjunction with global visual features.

9.1 Future Prospects

Reconstructions based on street level data provide great geometrical details of building facades but unfortunately lack completeness of the building. For instance, such reconstructions are missing roofs of buildings, courtyards or other parts of the building which are not visible from the street. On the other-hand aerial reconstructions provide great completeness of buildings, but lack geometric details (Fig. 9.1). In the future we did like to fuse the different reconstruction, to obtain complete and detailed model of urban environments. Such fusion comes with a great challenge of registering street-level images to aerial imagery. The registration process is particularly difficult due to the large viewpoint change between the two data sources, which makes patch based matching techniques brittle.

While those reconstruction stop at the entrance of a building, augmenting them with indoor reconstructions would provide the ultimate immersive environment. Here the challenge of aligning indoor with outdoor reconstructions is even bigger, since often very limited visual correspondences exist between the two models.

The proposed reconstruction algorithms make the strong assumption of capturing a static scene. This assumption often doesn't hold, especially when working with street level imagery, containing moving cars and pedestrians. Being able to detect and reconstruct moving objects would be the ultimate goal towards 3D reconstruction and scene understanding.

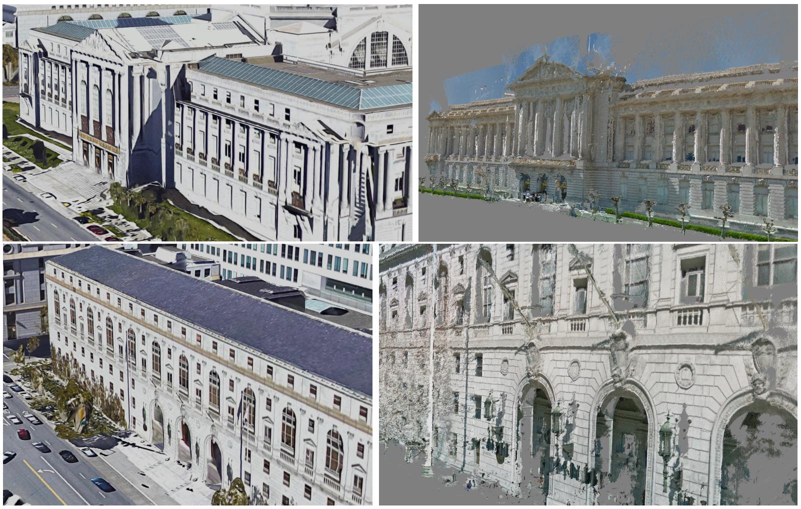


Figure 9.1: Left aerial reconstruction of the San Francisco cityhall (courtesy of google maps). Facades lack geometric details and are approximated as piecewise planar. Right, reconstruction of facade using street level imagery. The model is incomplete (missing roof) but provides detailed geometry.

Bibliography

[1] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Commun. ACM*, 54(10):105–112, Oct. 2011.

[2] S. Agarwal, K. Mierle, and Others. Ceres solver. `http://ceres-solver.org`.

[3] M. Agrawal. A lie algebraic approach for consistent pose registration for general euclidean motion. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 1891–1897, Oct. 2006.

[4] O. Ait-Aider, N. Andreff, J.-M. Lavest, and P. Martinet. Exploiting rolling shutter distortions for simultaneous object pose and velocity computation using a single view. In *IEEE International Conference on Computer Vision Systems (ICVS)*, pages 35–35, 2006.

[5] O. Ait-Aider, N. Andreff, J. M. Lavest, and P. Martinet. Simultaneous object pose and velocity computation using a single view from a rolling shutter camera. In *European Conference on Computer Vision (ECCV)*, pages 56–68. Springer, 2006.

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

[7] C. Albl, Z. Kukelova, and T. Pajdla. R6p - rolling shutter absolute camera pose. June 2015.

[8] J. B. Alexandre Karpenko, David E. Jacobs and M. Levoy. Digital video stabilization and rolling shutter correction using gyroscopes. Technical Report CTSR 2011-03, Stanford University, 2011.

[9] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer. Fast and incre-

mental method for loop-closure detection using bags of visual words. *Robotics, IEEE Transactions on*, 24(5):1027–1037, Oct. 2008.

[10] S. Baker, E. Bennett, S. B. Kang, and R. Szeliski. Removing rolling shutter wobble. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2392–2399, 2010.

[11] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics-TOG*, 28(3):24, 2009.

[12] J.-C. Bazin, C. Demonceaux, P. Vasseur, and I. Kweon. Motion estimation by decoupling rotation and translation in catadioptric vision. *Computer Vision and Image Understanding*, 114(2):254–273, 2010.

[13] J.-C. Bazin, C. Demonceaux, P. Vasseur, and I. Kweon. Rotation estimation and vanishing point extraction by omnidirectional vision in urban environment. *The International Journal of Robotics Research*, 31(1):63–81, 2012.

[14] J.-C. Bazin, H. Li, I. S. Kweon, C. Demonceaux, P. Vasseur, and K. Ikeuchi. A branch-and-bound approach to correspondence and grouping problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1565–1576, 2013.

[15] T. Boult. Remote Reality via Omnidirectional Imaging. In *DARPA Image Understanding Workshop (IUW)*, pages 1049–1052, Nov 1998.

[16] D. Bradley, B. Atcheson, I. Ihrke, and W. Heidrich. Synchronization and rolling shutter compensation for consumer video camera arrays. In *International Workshop on Projector-Camera Systems (PROCAMS)*, 2009.

[17] D. C. Brown. Close-range camera calibration. *Photogrammetric Engineering*, 37(8):855–866, 1971.

[18] S. Brugger. Secopdr self-contained pedestrain dead reckoning system for android mobile devices. 2012.

[19] R. T. Collins. A space-sweep approach to true multi-image matching. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*, pages 358–363. IEEE, 1996.

[20] P. Corke. An inertial and visual sensing system for a small autonomous helicopter. *Journal of Robotic Systems*, 21(2):43–51, 2004.

[21] J. M. Coughlan and A. L. Yuille. Manhattan world: Compass direction from a single image by bayesian inference. In *Internation Conference on Computer Vision*, pages 941–947, 1999.

[22] D. A. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, 3/e (Undergraduate Texts in Mathematics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[23] M. Cummins and P. Newman. FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008.

[24] J. Domke and Y. Aloimonos. Integration of visual and inertial information for egomotion: a stochastic approach. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2053–2059. IEEE, 2006.

[25] D. Feldman, T. Pajdla, and D. Weinshall. On the epipolar geometry of the crossed-slits projection. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 988–995 vol.2, Oct.

[26] 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.

[27] P.-E. Forssén and E. Ringaby. Rectifying rolling shutter video from hand-held devices. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2010.

[28] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys. Building rome on a cloudless day. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 368–381, Berlin, Heidelberg, 2010. Springer-Verlag.

[29] 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 *ECCV (4)*, pages 269–282, 2010.

[30] F. Fraundorfer, C. Wu, and M. Pollefeys. Combining monocular and stereo cues for mobile robot localization using visual words. pages 1–4, 2010.

[31] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Manhattan-world stereo. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Miami, FL, June 2009. IEEE.

[32] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Reconstructing building interiors from images. In *Proc. of the International Conference on Computer Vision (ICCV)*, 2009.

[33] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2010.

[34] A. Fusiello, E. Trucco, and A. Verri. A compact algorithm for rectification of stereo pairs. *Mach. Vis. Appl.*, 12(1):16–22, 2000.

[35] S. Galliani, K. Lasinger, and K. Schindler. Massively parallel multiview stereopsis by surface normal diffusion.

[36] D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *Proc. CVPR*, pages 1–8, June.

[37] C. Geyer, M. Meingast, , and S. Sastry. Geometric models of rolling-shutter cameras. In *Proceedings of OMNIVIS*, 2005.

[38] Google. Street view, 2009.

[39] M. Grundmann, V. Kwatra, D. Castro, and I. Essa. Effective calibration free rolling shutter removal. *IEEE International Conference on Computational Photography (ICCP)*, 2012.

[40] R. Gupta and R. I. Hartley. Linear pushbroom cameras. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(9):963–975, Sept. 1997.

[41] G. Hanning, N. Forslöw, P.-E. Forssén, E. Ringaby, D. Törnqvist, and J. Callmer. Stabilizing cell phone video using inertial measurement sensors. In *The Second IEEE International Workshop on Mobile Vision*, 2011.

[42] R. Haralick, D. Lee, K. Ottenburg, and M. Nolle. Analysis and solutions of the three point perspective pose estimation problem. In *Computer Vision and Pattern Recognition (CVPR)*, 1991.

[43] R. M. Haralick, C.-N. Lee, K. Ottenberg, and M. Nölle. Review and analysis of solutions of the three point perspective pose estimation problem. *Int. J. Comput. Vision*, 13(3):331–356, Dec. 1994.

[44] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[45] J. Hedborg, P.-E. Forssén, M. Felsberg, and E. Ringaby. Rolling shutter bundle adjustment. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[46] J. Heinly, J. L. Schonberger, E. Dunn, and J.-M. Frahm. Reconstructing the world* in six days*(as captured by the yahoo 100 million image dataset). In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3287–3295, 2015.

[47] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(2):328–341, Feb. 2008.

[48] A. Jordt-Sedlazeck, D. Jung, and R. Koch. Refractive plane sweep for underwater images. In *GCPR 2013*, 2013.

[49] 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, 2011.

[50] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *Proc. Eigth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, 2009.

[51] B. Klingner, D. Martin, and J. Roseborough. Street view motion-from-structure-from-motion. In *Proceedings of the International Conference on Computer Vision*, 2013.

[52] L. Kneip, A. Martinelli, S. Weiss, D. Scaramuzza, and R. Siegwart. Closed-form solution for absolute scale velocity determination combining inertial measurements and a single feature correspondence. In *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011*, pages 4546–4553, 2011.

[53] Z. Kukelova, M. Bujnak, and T. Pajdla. Automatic generator of minimal problem solvers. In *European Conference on Computer Vision (ECCV)*, pages 302–315. Springer, 2008.

[54] D. C. Lee, M. Hebert, and T. Kanade. Geometric reasoning for single image structure recovery. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[55] G. H. Lee, F. Fraundorfer, and M. Pollefeys. Mav visual slam with plane constraint. In *ICRA*, pages 3139–3144, 2011.

[56] G. H. Lee, B. Li, M. Pollefeys, and F. Fraundorfer. Minimal solutions for pose estimation of a multi-camera system. In *International Symposium on Robotics Research (ISRR)*, 2013.

[57] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate o(n) solution to the pnp problem. In *International Journal of Computer Vision (IJCV)*, volume 81, 2009.

[58] A. Levin and R. Szeliski. Visual odometry and map correlation. pages I: 611–618, 2004.

[59] M. Lhuillier. Fusion of gps and structure-from-motion using constrained bundle adjustments. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3025–3032. IEEE, 2011.

[60] B. Li, L. Heng, G. H. Lee, and M. Pollefeys. A 4-point algorithm for relative pose estimation of a calibrated camera with a known relative rotation angle. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS 2013, Tokyo, Japan*, 2013.

[61] X. Li, C. Wu, C. Zach, S. Lazebnik, and J.-M. Frahm. Modeling and recognition of landmark image collections using iconic scene graphs. In *Proceedings of the 10th European Conference on Computer Vision:*

Part I, ECCV '08, pages 427–440, Berlin, Heidelberg, 2008. Springer-Verlag.

[62] C.-K. Liang, L.-W. Chang, and H. Chen. Analysis and compensation of rolling shutter effect. *IEEE Transactions on Image Processing*, 17(8):1323–1330, Aug.

[63] A. Lippman. Movie-maps: An application of the optical videodisc to computer graphics. *SIGGRAPH Comput. Graph.*, 14(3):32–42, 1980.

[64] F. Liu, M. Gleicher, J. Wang, H. Jin, and A. Agarwala. Subspace video stabilization. *ACM Trans. Graph.*, 30(1):4:1–4:10, Feb. 2011.

[65] J. Lobo and J. Dias. Vision and inertial sensor cooperation using gravity as a vertical reference. 25(12):1597–1608, December 2003.

[66] P. Lothe, S. Bourgeois, F. Dekeyser, E. Royer, and M. Dhome. Towards geographical referencing of monocular slam reconstruction using 3d city models: Application to real-time accurate vision-based localization. pages 2882–2889, 2009.

[67] 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.

[68] D. G. Lowe. Distinctive image features from scale-invariant keypoints. In *International Journal on Computer Vision (IJCV)*, volume 60, pages 91–110, 2004.

[69] D. F. M. Rothermel, K. Wenzel and N. Haala. Sure: Photogrammetric surface reconstruction from imagery. In *Proceedings LC3D Workshop*, 2014.

[70] 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 *European Conference on Computer Vision (ECCV)*, pages 456–469. Springer, 2012.

[71] E. Malis and M. Vargas. Deeper understanding of the homography decomposition for vision-based control. Research Report RR-6303, 2007.

[72] A. Martinelli. Closed-form solution of visual-inertial structure from motion. *International Journal of Computer Vision*, 106(2):138–152, 2014.

[73] M. Meilland, T. Drummond, and A. I. Comport. A unified rolling shutter and motion blur model for 3d visual registration. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2016–2023, 2013.

[74] F. Moreno-Noguer, V. Lepetit, and P. Fua. Accurate non-iterative o(n) solution to the pnp problem. In *International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.

[75] A. C. Murillo and J. Kosecka. Experiments in place recognition using gist panoramas. In *IEEE Workshop on Omnidirectional Vision, Camera Netwoks and Non-Classical Cameras, ICCV 2009*, pages 1–8, 2009.

[76] O. Naroditsky, X. S. Zhou, J. H. Gallier, S. I. Roumeliotis, and K. Daniilidis. Two efficient solutions for visual odometry using directional correspondence. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(4):818–824, 2012.

[77] R. A. Newcombe and A. J. Davison. Live dense reconstruction with a single moving camera. In *CVPR*, pages 1498–1505. IEEE, 2010.

[78] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(6):756–777, 2004.

[79] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. pages 2161–2168, 2006.

[80] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. 42(3):145–175, May 2001.

[81] O. Oreifej, N. da Vitoria Lobo, and M. Shah. Horizon constraint for unambiguous uav navigation in planar scenes. In *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011*, pages 1159–1165, 2011.

[82] D. Ortin and J. Montiel. Indoor robot motion based on monocular images. *Robotica*, 19(3):331–342, 2001.

[83] L. Oth, P. Furgale, L. Kneip, and R. Siegwart. Rolling shutter camera calibration. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[84] M. Pollefeys, D. Nistér, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.-J. Kim, P. Merrell, et al. Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision*, 78(2-3):143–167, 2008.

[85] M. Pollefeys, D. Nistér, J. M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, and H. Towles. Detailed real-time urban 3d reconstruction from video. *Int. J. Comput. Vision*, 78(2-3):143–167, July 2008.

[86] A. Pronobis, M. Fornoni, H. I. Christensen, and B. Caputo. The robot vision task at imageclef 2010. In *In the Working Notes of CLEF 2010, Padova, Italy*, 2010.

[87] L. Quan and Z. D. Lan. Linear n-point camera pose determination. In *Pattern Analysis and Machine Intelligence (PAMI)*, volume 21, pages 774–780, 1999.

[88] E. Ringaby and P.-E. Forssén. Efficient video rectification and stabilisation for cell-phones. *International Journal of Computer Vision (IJCV)*, 96(3):335–352, 2012.

[89] O. Saurer, F. Fraundorfer, and M. Pollefeys. Omnitour: Semi-automatic generation of interactive virtual tours from omnidirectional video. In *International Symposium on 3D Data Processing, Visualization and Transmission*, volume 139, 2010.

[90] O. Saurer, F. Fraundorfer, and M. Pollefeys. Visual localization using global visual features and vanishing points. In *CLEF (Notebook Papers/LABs/Workshops)*, volume 10, 2010.

[91] O. Saurer, F. Fraundorfer, and M. Pollefeys. Homography based visual odometry with known vertical direction and weak manhattan world assumption. In *Vicomor Workshop at IROS*, 2012.

[92] O. Saurer, K. Koser, J.-Y. Bouguet, and M. Pollefeys. Rolling shutter

stereo. In *IEEE International Conference on Computer Vision (ICCV)*, pages 465–472, 2013.

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

[94] O. Saurer, P. Vasseur, C. Demonceaux, and F. Fraundorfer. A homography formulation to the 3pt plus a common direction relative pose problem. In *Computer vision–ACCV*. Springer, 2014.

[95] D. Scaramuzza. 1-point-ransac structure from motion for vehicle-mounted cameras by exploiting non-holonomic constraints. *International Journal of Computer Vision*, 95(1):74–85, 2011.

[96] D. Scaramuzza, F. Fraundorfer, and R. Siegwart. Real-time monocular visual odometry for on-road vehicles with 1-point ransac. In *2009 IEEE International Conference on Robotics and Automation*, pages 1–7, 2009.

[97] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. pages 1–7, 2007.

[98] P. Schmid. Mobiletour: Indoor exploration system for android. 2011.

[99] J. L. Schönberger, F. Fraundorfer, and J.-M. Frahm. Structure-from-motion for mav image sequence analysis with photogrammetric applications. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-3:305–312, 2014.

[100] P. Smith, T. Drummond, and K. Roussopoulos. Computing map trajectories by representing, propagating and combining pdfs over groups. pages 1275–1282, 2003.

[101] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[102] J.-P. Tardif, Y. Pavlidis, and K. Daniilidis. Monocular visual odometry in urban environments using an omnidirectional camera. In *IROS*, pages 2531–2538, 2008.

[103] C. Taylor. Videoplus: a method for capturing the structure and appearance of immersive environments. In *IEEE Trans. Visual. Comp. Graphics*, pages 171–182, April-June 2002.

[104] G. Thalin. Camera rolling shutter amounts.

[105] A. B. Torralba, K. P. Murphy, W. T. Freeman, and M. A. Rubin. Context-based vision system for place and object recognition. In *ICCV*, pages 273–280, 2003.

[106] B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ICCV, pages 298–372. Springer, 2000.

[107] C. Troiani, S. Al Zanati, and A. Martinelli. A 3 Points Vision Based Approach for MAV Localization in GPS Denied Environments. In *6th European Conference on Mobile Robots*, Barcelona, Spain, 2013.

[108] C. Troiani, A. Martinelli, C. Laugier, and D. Scaramuzza. 1-point-based monocular motion estimation for computationally-limited micro aerial vehicles. In *Mobile Robots (ECMR), 2013 European Conference on*, pages 13–18. IEEE, 2013.

[109] C. Troiani, A. Martinelli, C. Laugier, and D. Scaramuzza. 2-point-based outlier rejection for camera-imu systems with applications to micro aerial vehicles. In *IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, 2014.*, 2014.

[110] A. Uyttendaele, A. Criminisi, S. B. Kang, S. Winder, R. Hartley, and S. Richard. High-quality image-based interactive exploration of real-world environments. Technical Report MSR-TR-2003-61, Microsoft Research, 2003.

[111] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

[112] J. Ventura, C. Arth, G. Reitmayr, and D. Schmalstieg. A minimal solution to the generalized pose-and-scale problem. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[113] T. Viéville, E. Clergue, and P. D. S. Facao. Computation of ego-motion and structure from visual and inertial sensors using the vertical cue. In

Computer Vision, 1993. Proceedings., Fourth International Conference on, pages 591–598. IEEE, 1993.

[114] S. Weiss and R. Siegwart. Real-time metric state estimation for modular vision-inertial systems. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4531–4537. IEEE, 2011.

[115] R. Yang and M. Pollefeys. Multi-resolution real-time stereo on commodity graphics hardware. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–211. IEEE, 2003.

[116] R. Yang and M. Pollefeys. Multi-resolution real-time stereo on commodity graphics hardware. In *Proceedings CVPR*, volume 1, pages I–211–I–217 vol.1, 2003.

[117] A. Zomet, D. Feldman, S. Peleg, and D. Weinshall. Mosaicing new views: the crossed-slits projection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(6):741–754, 2003.