Self-calibration for consumer omnidirectional multi-camera mounted on a helmet

Thanh-Tin Nguyen

▶ **To cite this version:**

Thanh-Tin Nguyen. Self-calibration for consumer omnidirectional multi-camera mounted on a helmet. Automatic. Université Clermont Auvergne [2017-2020], 2017. English. NNT : 2017CLFAC060 . tel-01763861

HAL Id: tel-01763861
https://theses.hal.science/tel-01763861

Submitted on 11 Apr 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

EDSPIC : 831

Université Clermont Auvergne

Ecole Doctorale
Sciences Pour L'Ingénieur De Clermont-Ferrand

Thèse

présentée par:

Thanh-Tin Nguyen

pour obtenir le grade de

Docteur d'Université

Spécialité : Vision pour la robotique

Auto-calibration d'une multi-caméra omnidirectionnelle grand public fixée sur un casque

Self-calibration for consumer omnidirectional multi-camera mounted on a helmet

---

Soutenue publiquement le 19 décembre 2017 devant le jury :

M.	Michel	DHOME	DR CNRS Institut Pascal	Président du jury
M.	David	FOFI	Prof. Univ. de Bourgogne	Rapporteur
M.	Maxime	LHUILLIER	CR CNRS Insitut Pascal	Directeur de thèse
Mme.	Sylvie	NAUDET-COLLETTE	Chercheuse, CEA Saclay	Examinatrice
M.	Pascal	VASSEUR	Prof. Univ. de Rouen	Rapporteur

Acknowledgements

First of all, I would like to express my thanks to Pascal Vasseur and David Fofi for having reviewed this works. I am grateful to jury members, namely Michel Dhome and Sylvie Naudet-Collette for your times and positive comments to improve my works.

Foremost, I would like to express my sincere gratitude to my supervisor Maxime Lhuillier for his enduring support and inspiration, for his patience, enthusiasm, without his continuous guidance I would never able to get so far.

I would like to acknowledge CNRS, Université Clermont Auvergne and Institut Pascal for funding. I do not forget to thank our team leader Thierry Chateau and Céline Teulière for the perfect working and researching environment and their helps during my thesis. My thanks go to all members of the laboratory, in particular to Datta Ramadasan for his information advices in C++, my colleagues in the office: Francois De la Bourdonnaye, Yosra Dorai, Mhalla Ala and the other colleagues in ComSee team for their helps.

Finally, I would like to thank my friends and my family for their support and encouragements. Thank you very much!

Abstract

360 degree and spherical multi-cameras built by fixing together several consumer cameras become popular and are convenient for recent applications like immersive videos, 3D modeling and virtual reality. This type of cameras allows to include the whole scene in a single view. When the goal of our applications is to merge monocular videos together into one cylinder video or to obtain 3D informations from environment, there are several basic steps that should be performed beforehand. Among these tasks, we consider the synchronization between cameras; the calibration of multi-camera system including intrinsic and extrinsic parameters (i.e. the relative poses between cameras); and the rolling shutter calibration. The goal of this thesis is to develop and apply user friendly method. Our approach does not require a calibration pattern.

First, the multi-camera is initialized thanks to assumptions that are suitable to an omnidirectional camera without a privileged direction: the cameras have the same setting (frequency, image resolution, field-of-view) and are roughly equiangular. Second, a frame-accurate synchronization is estimated from instantaneous angular velocities of each camera provided by monocular Structure-from-Motion. Third, both inter-camera poses and intrinsic parameters are refined using multi-camera Structure-from-Motion and bundle adjustment. Last, we introduce a bundle adjustment that estimates not only the usual parameters but also a subframe-accurate synchronization and the rolling shutter.

We experiment in a context that we believe useful for applications (3D modeling and 360 videos): several consumer cameras or a spherical camera mounted on a helmet and moving along trajectories of several hundreds of meters or kilometers.

Keywords: Bundle adjustment, self-calibration, synchronization, rolling shutter, multi-camera.

Résumé

Les caméras sphériques et 360 deviennent populaires et sont utilisées notamment pour la création de vidéos immersives et la génération de contenu pour la réalité virtuelle. Elles sont souvent composées de plusieurs caméras grand-angles/fisheyes pointant dans différentes directions et rigidement liées les unes aux autres. Cependant, il n'est pas si simple de les calibrer complètement car ces caméras grand public sont rolling shutter et peuvent être mal synchronisées. Cette thèse propose des méthodes permettant de calibrer ces multi-caméras à partir de vidéos sans utiliser de mire de calibration.

On initialise d'abord un modèle de multi-caméra grâce à des hypothèses appropriées à un capteur omnidirectionnel sans direction privilégiée : les caméras ont les mêmes réglages (dont la fréquence et l'angle de champs de vue) et sont approximativement équiangulaires. Deuxièmement, sachant que le module de la vitesse angulaire est le même pour deux caméras au même instant, nous proposons de synchroniser les caméras à une image près à partir des vitesses angulaires estimées par structure-from-motion monoculaire. Troisièmement, les poses inter-caméras et les paramètres intrinsèques sont estimés par structure-from-motion et ajustement de faisceaux multi-caméras avec les approximations suivantes: la multi-caméra est centrale, global shutter; et la synchronisation précédant est imposée. Enfin, nous proposons un ajustement de faisceaux final sans ces approximations, qui raffine notamment la synchronisation (à précision sous-trame), le coefficient de rolling shutter et les autres paramètres (intrinsèques, extrinsèques, 3D).

On expérimente dans un contexte que nous pensons utile pour des applications comme les vidéos 360 et la modélisation 3D de scènes : plusieurs caméras grand public ou une caméra sphérique fixée(s) sur un casque et se déplaçant le long d'une trajectoire de quelques centaines de mètres à quelques kilomètres.

Mot clés : ajustement de faisceaux, auto-étalonnage, synchronisation, rolling shutter, multi-caméra.

Contents

Abbreviations

LM	Levenberg-Marquardt method
IAV	Instantaneous Angular Velocity
FoV	Field of View
DoF	Degree of Freedom
FpS	Frame per Second
FA	Frame Accurate
SFA	Sub-Frame Accurate
GS	Global Shutter
RS	Rolling Shutter
BA	Bundle Adjustment
MCBA	Multi-Camera Bundle Adjustment
SfM	Structure from Motion
SLAM	Simultaneous Localization And Mapping
SLERP (*Slerp*)	Special Linear intERPolation
SQUAD (*Squad*)	Spherical and QUADrangle
IMU	Inertial Measurement Unit

List of Figures

List of Tables

Chapter 1

Introduction

Numerous applications can be realized on 2D images taken by a digital camera with the help of a computer. Omnidirectional cameras, among others, are found in many applications such as surveillance, navigation, robotics, 360 video, virtual reality, telepresence, etc. Several applications require a 3D model of an environment which is a classical problem in the domain of Computer Vision. For example, the efficient modeling of a complete environment requires an acquisition on (almost) the whole viewing sphere. In comparison to scanners (Lidar, Time-of-flight camera, structured light), the cameras are usually less costly and more easily to be embedded. For these reasons, our interest is focused on omnidirectional "multi-camera" sensors built by fixing together several consumer cameras pointing in different directions. See Figure 1.1 for examples of omnidirectional multi-cameras or 360 cameras [360rize], [ThetaS], [Gear360], [Virb360], [Ladybug2]. These cameras become popular thanks to their prices, high resolutions, growing applications including 360 videos (e.g. in YouTube), generation of virtual reality content, 3D scene modeling. The advantage of omnidirectional multi-cameras is primarily that they have a large field-of-view and thus provide a large part of the surrounding environment in one instant. They allow to establish more spacious point correspondences which lead to more complete 3D reconstruction and more stable egomotion estimation than standard cameras. Various combinations of several monocular cameras lead to various types of omnidirectional multi-cameras. Sometimes it arises from physical setup or from types of optical elements (lenses, camera sensors, etc) or from the acquisition process.

Camera can be understood as ray-based sensing device. We say that a multi-camera is central, or has a single viewpoint or a projection center, if all its rays intersect at a single point. Otherwise, the baseline defined by the distance between the centers of two cameras is not zero. Depending on the baseline, the multi-camera systems divide into *central* and *non-central* ones.

The digital cameras map surrounding space through optical systems onto photosensitive devices (e.g. charge coupled device - CCD, complementary metal oxide semiconductor - CMOS). Today, most digital cameras use a CMOS sensor because of its lower consumption, cheaper manufacturing and the potential for on-chip processing. What sensor is used leads to two methods of image capture: *rolling shutter* camera (using CMOS sensor) captures the image row-by-row and *global shutter* camera (using CCD sensor) captures the entire frame at the same time.

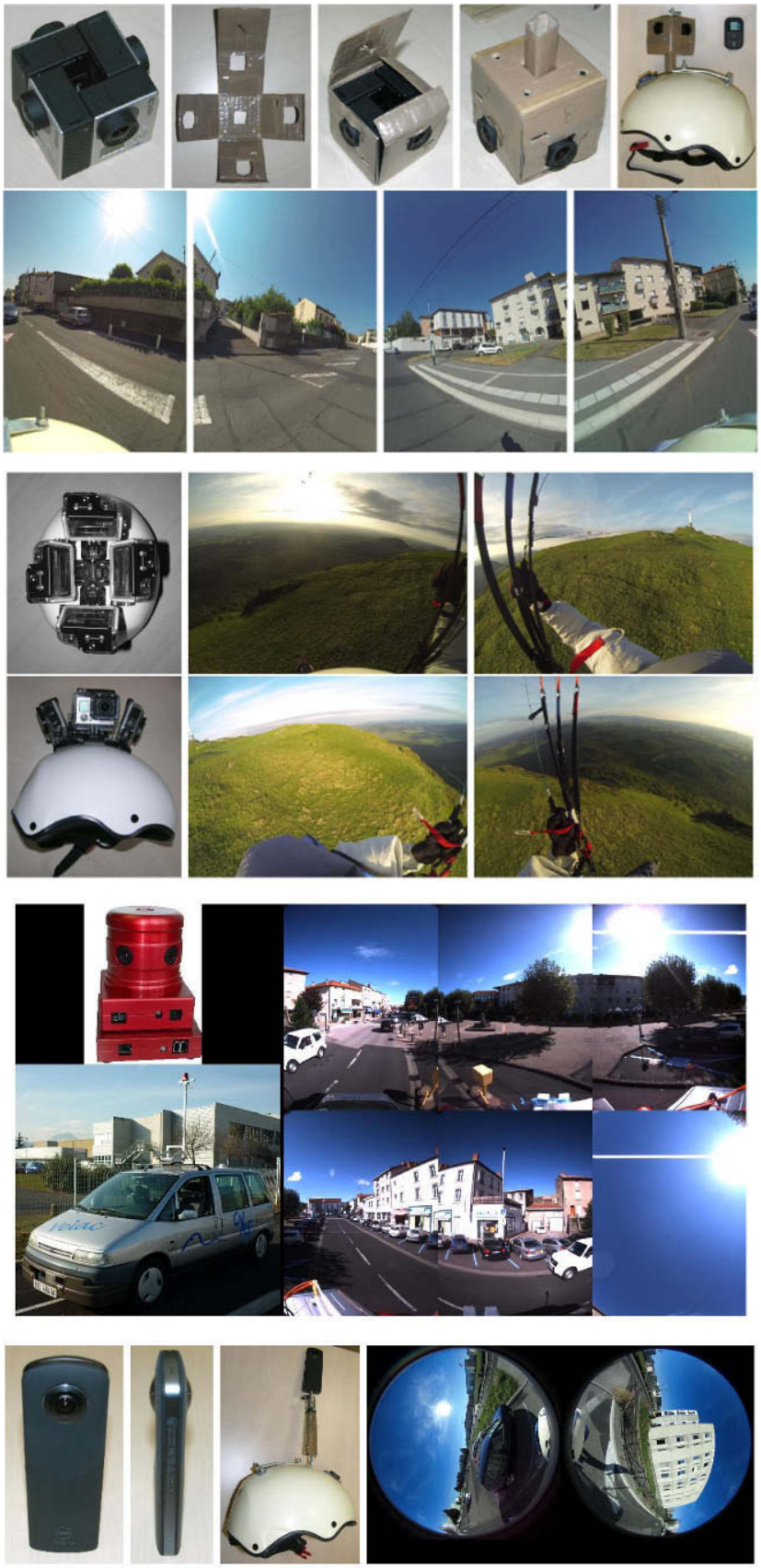


FIGURE 1.1: Multi-camera systems and acquired images. From the top: four Gopro Hero 3 in a cardboard; four Gopro Hero 3 in their housing; PointGrey Ladybug 2 and Ricoh Theta S.

The process of recovering the temporal relationship between two or more videos is necessary for successfully analyzing and integrating available information from several cameras. We assume that each video has a constant frame rate. The synchronization methods can be classified in two categories depending on the goal of estimating *frame-accurate* or *subframe-accurate* time-offsets. In the former, frames with the same index are taken at the same time up to inverse of frame per second. Moreover, there is in general also a subframe offset. Using the results of the former, the synchronization problem can be reformulated to achieve the latter.

This thesis deals with *omnidirectional multi-cameras*. Our work investigates their properties, synchronization and self-calibration including rolling shutter. As the main contribution, a complete self-calibration method including synchronization and rolling shutter is developed.

1.1 Motivation

This work is motivated by using omnidirectional multi-cameras as in Figure 1.1 in applications such as 360 video (e.g. on YouTube) and 3D scene modeling. We do assumptions that are suitable to an omnidirectional multi-camera without a privileged direction: the monocular cameras have the same setting (frequency, image resolutions, field of view, etc). Applications using multiple camera rigs require accurate calibration of the device, which includes intrinsic parameters and relative poses of each camera with respect to multi-camera coordinate system. Our work concentrates on the uncalibrated case without assumptions about the scene (except for rigidity), and establishes relative pose of multi-camera rigs even if the cameras do not have overlapping view. However, their complete self-calibration is not easy since the consumer fisheyes are rolling shutter cameras which can be unsynchronized.

360 video needs a central multi-camera. The smaller the baseline, the better the stitching quality. The user/manufacturer can reduce the baseline (and the multi-camera price) thanks to a small number of cameras. Here, we mainly use a DIY[1] multi-camera composed of four Gopro Hero 3 cameras enclosed in a cardboard (Figure 1.1) such that the baseline is as small as possible. Since a small number of cameras also reduces the field-of-view (FoV) shared by adjacent cameras, we avoid methods that rely on this shared FoV such as image matching between different monocular videos. A greater number of cameras can be used in order to increase the overlapping FoV, but both price and baseline increase [Virb360]. We also experiment using a spherical camera (Figure 1.1) having only two large FoV (fisheye-like) images.

Moreover, the video sequence synchronization is essential step for the calibration of multiple cameras and for any Computer Vision application. In professional applications, a hardware synchronization usually offers high accuracy requiring a physical connection but is expensive and complex. In many cases like Gopro cameras, the manufacturer provides a wifi-based synchronization: the user starts all videos at once by a single click. However, the resulting time offsets between videos are too inaccurate for applications: about 0.04 seconds and sometimes above 0.1 seconds in our experiments. Assume that a central multi-camera moves at 20km/h (e.g. biking in a city) and two cameras have a time offset equal to only 0.02 second, then explain consequences on a 360 video obtained

[1] Do It Yourself

by video stitching. If we neglect this offset, the two videos are stitched as if they have same camera centers at same frame number, although the distance between these centers is 0.11 meter (20/3.6*0.02). This generates artifacts in the 360 video due to foreground objects that are in the FoV shared by the two cameras. Therefore, the synchronization problem should be solved after video acquisition.

The low price of a consumer camera implies that the camera is rolling shutter. This means that two different lines of pixels of a frame are acquired at different instants. This produces predictable distortions in the image when either the camera or the objects in the scene are moving. In a global shutter, all pixels have the same time. Most Computer Vision algorithms assume that the cameras have a global shutter although they do not. This can degrade the quality of results in applications such as 3D reconstruction, similarly as inaccurate synchronization degrades the quality of 360 videos.

1.2 Goals of the thesis

The goal of the thesis is to develop a self-calibration method of omnidirectional multi-camera. Our self-calibration takes into account challenges: lack of synchronization, rolling shutter. The main goals are:

- to synchronize two or more cameras with frame accuracy and subframe accuracy,
- to self-calibrate omnidirectional multi-camera rig from images without assuming overlapping view between cameras,
- to estimate the coefficient of rolling shutter (time delay between two successive lines) and the time offsets with subframe accuracy between cameras.

We propose a purely visual approach without additional sensors.

1.3 Contribution of the thesis

The following contributions are brought by the thesis

1. A robust synchronization method based on instantaneous angular velocity assuming jointly moving cameras,
2. Improvements of a previous bundle adjustment designed for global shutter multi-cameras (refines intrinsic, extrinsic parameters and the other 3D parameters),
3. The first multi-camera bundle adjustment that estimates not only rolling shutter (line delay) but also synchronization (time offsets), in addition to the usual 3D parameters,
4. Experiments on long trajectories (hundreds of meters or kilometers) in a context that we believe useful for applications (3D modeling and 360 video): several consumer cameras or a spherical camera mounted on a helmet and moving along trajectories by walking or biking.

1.4 Structure of the thesis

In Chapter 2, the basic terminology used throughout the thesis and overview of theoretical preliminaries are explained. In particular, we present omnidirectional projection, parameterizations of rotation, least-square optimizations. Furthermore, we briefly review the structure-from-motion algorithm that we use. In Chapter 3, previous and related works are reviewed. The three next chapters explain our works.

In Chapter 4, two monocular camera models (the classical polynomial distortion model and the unified camera model) are presented. We tackle monocular camera as follows: the intrinsic parameters are initialized thanks to assumptions that are suitable to an omnidirectional camera without a privileged direction. A frame accurate synchronization between all videos is obtained using a method based on instantaneous angular velocity.

In Chapter 5, a central multi-camera calibration is initialized from the estimated intrinsic monocular parameters and approximate inter-camera rotations. We apply multi-camera structure-from-motion and improve a previous multi-camera bundle adjustment in order to optimize the intrinsic and extrinsic parameters, the multi-camera pose and 3D structure. Up to now, we do assume that the cameras are global shutter and the time offsets between cameras are integers (frame accurate).

In Chapter 6, we introduce a novel multi-camera bundle adjustment for estimating subframe-accurate synchronization and line delay rolling shutter parameter in addition to usual parameters. We experiment using videos taken by multi-cameras mounted on a helmet and moving along trajectories of several hundreds of meters or kilometer, then compare our self-calibration results with ground truth.

Finally, Chapter 7 concludes the thesis.

Chapter 2

Preliminaries

In this chapter, we present notation conventions and outline some important concepts used throughout this work. A model allowing to represent omnidirectional multi-camera and all scene points around the camera is described. We briefly present the optimization framework, especially Levenberg-Marquardt method which is used to solve our non-linear least squares problems. We also summarize an efficient framework for joint estimation of camera positions and 3D structure scene from a moving calibrated camera. This framework is called Structure from Motion. One strategy used in this dissertation is investigated in [Mouragnon+09].

Contents

2.1 Omnidirectional projection

The camera is a mapping from the 3D world (Euclidean 3-space $\mathbb{R}^3$) to a 2D image (Euclidean 2-space $\mathbb{R}^2$) in which we lose one dimension. This process is usually modeled by central projection in which projecting rays from 3D world points pass through a fixed point in space, called as the *center of projection*. This ray intersects a plane in space, called as the *image plane* or focal plane, at an image 2D point. *Omnidirectional cameras* have various FoV, e.g. *fish-eye camera* with FoV larger than 180^o, *catadioptric camera* consisting of curved mirrors in front of camera with 360^o in the horizontal direction and various FoV in the vertical direction, etc. *Multi-camera rigs*, as in Figure 1.1, are

omnidirectional cameras as well. In this section, we start with the simplest camera model and then describe a rigid multi-camera system.

The basic pinhole model. Assume that the center of projection is at the origin of an Euclidean coordinate system and the plane $Z = f$ is the image plane. The center of projection is called the *camera center*. The line from the camera center perpendicular to the image plane is called the *principal axis* or *principal ray* of the camera. The point where the principal axis intersects the image plane is called the *principal point*. A point $\mathbf{X}_\mathrm{C}$ in 3D space is mapped to a point on the image plane that is the intersection of the image plane and the line joining the point $\mathbf{X}_\mathrm{C}$ to the camera center. If $\mathbf{X}_\mathrm{C}$ is represented by the homogeneous 4-vector $(X_\mathrm{C}, Y_\mathrm{C}, Z_\mathrm{C}, w_\mathrm{C})^\top$, the image point $\mathbf{x}$ is represented by a homogeneous 3-vector, the projection function is expressed conveniently in homogeneous coordinates as:

$$\mathbf{x} = \mathtt{K}[\mathtt{I} \mid \mathbf{0}]\mathbf{X}_\mathrm{C} \tag{2.1}$$

where the camera calibration matrix is

$$\mathtt{K} = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.2}$$

The calibration matrix has focal parameters (f_x, f_y), principal point $\mathbf{p}_0 = (u_0, v_0)^\top$ and skew term s which models non-rectangular pixels. The homogeneous vector $\mathbf{x} = (X, Y, Z)^\top$ with $Z \neq 0$ represents the image point $\mathbf{p} = (u, v)^\top$ with inhomogeneous coordinate

$$\begin{pmatrix} u \\ v \end{pmatrix} = \pi(\mathbf{x}) = \begin{pmatrix} X/Z \\ Y/Z \end{pmatrix}. \tag{2.3}$$

The camera is assumed to be located at the origin of an Euclidean coordinate system with the principal axis of the camera pointing straight down the $Z_\mathrm{C}-$axis, and the point $\mathbf{X}_\mathrm{C}$ is expressed in this coordinate system. Such a coordinate system is called the *camera coordinate system* (see Figure 2.1(a)).

Other models. The above model is used for perspective camera. Many different cameras consisting of omnidirectional cameras have been investigated and a certain number of computational models for cameras are proposed in the literature. A survey of camera models existing in the Computer Vision can be found in [Sturm+11]. Generally, we add various terms of radial, tangential and other distortions to the basic pinhole model in order to model more precisely for cameras with a large or very wide FoV. Radial distortion models have different forms and are defined by distortion function that links radial distance (distance between image point and distortion center) in the distorted image to either radial distance in the undistorted image or the incidence angle between camera ray and principal ray. We also remind the equiangular case where the incidence angle between camera ray and principal ray is proportional to radial distance in the distorted image. The camera models that we use, will be described in Chapter 4.

Camera rotation and translation. In general, points in space are expressed in terms of a different Euclidean coordinate system, known as the *world coordinate system*. The transformation between two coordinate systems is represented by a rotation $\mathtt{R}$ and a translation $\mathbf{t}$. If inhomogeneous vectors $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{X}}_\mathrm{C}$ represent the same point in the world coordinate system and the camera coordinate system respectively, then we write

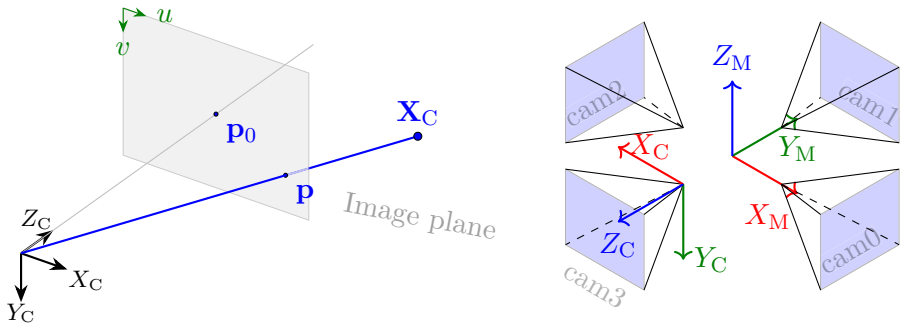
$\tilde{\mathbf{X}} = \mathtt{R}\tilde{\mathbf{X}}_\mathrm{C} + \mathbf{t}$. This equation may be rewritten in homogeneous coordinates as

$$\mathbf{X}_\mathrm{C} = \begin{bmatrix} \mathtt{R}^\top & \mathtt{R}^\top\mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{X}. \tag{2.4}$$

Putting this together with Eq.(2.1) leads to the formula:

$$\mathbf{x} = \mathtt{K}\mathtt{R}^\top[\mathtt{I} \mid \quad \mathbf{t}]\mathbf{X}. \tag{2.5}$$

where $\mathtt{P} = \mathtt{K}\mathtt{R}^\top[\mathtt{I} \mid \quad \mathbf{t}]$ is the *projection matrix* and $\mathbf{X}$ is in the world coordinate system. This is the general mapping given by a pinhole camera.



(a) Perspective camera model (b) Multi-camera system

FIGURE 2.1: Perspective camera model and multi-camera system.

Rigid multi-camera system modeling. For a rigid multi-camera system, we also define a *multi-camera coordinate system* (see Figure 2.1(b)). The world coordinate system and the multi-camera coordinate system are related via a rotation $\mathtt{R}_\mathrm{M}$ and a translation $\mathbf{t}_\mathrm{M}$. The transformation between the multi-camera coordinate system and the camera coordinate system is also represented by a rotation $\mathtt{R}_\mathrm{C}$ and a translation $\mathbf{t}_\mathrm{C}$. If the monocular projection centers coincide, the multi-camera is *central* and $\mathbf{t}_\mathrm{C} = \mathbf{0}$. The multi-camera is, in general, *non central*. If a 3D point is represented by inhomogeneous vectors $\tilde{\mathbf{X}}$, $\tilde{\mathbf{X}}_\mathrm{M}$ and $\tilde{\mathbf{X}}_\mathrm{C}$ in the world coordinate system, the multi-camera coordinate system and the camera coordinate system, respectively, then we can write

$$\tilde{\mathbf{X}} = \mathtt{R}_\mathrm{M}\tilde{\mathbf{X}}_\mathrm{M} + \mathbf{t}_\mathrm{M} \text{ and } \tilde{\mathbf{X}}_\mathrm{M} = \mathtt{R}_\mathrm{C}\tilde{\mathbf{X}}_\mathrm{C} + \mathbf{t}_\mathrm{C}. \tag{2.6}$$

Thus

$$\tilde{\mathbf{X}}_\mathrm{C} = \mathtt{R}_\mathrm{C}^\top \left(\mathtt{R}_\mathrm{M}^\top (\tilde{\mathbf{X}} \quad \mathbf{t}_\mathrm{M}) \quad \mathbf{t}_\mathrm{C} \right). \tag{2.7}$$

This transformation can be represented by a linear transformation of homogeneous coordinates

$$\begin{aligned} \mathbf{X}_\mathrm{C} &= \begin{bmatrix} \mathtt{R}_\mathrm{C}^\top & \mathtt{R}_\mathrm{C}^\top\mathbf{t}_\mathrm{C} \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{X}_\mathrm{M} = \begin{bmatrix} \mathtt{R}_\mathrm{C}^\top & \mathtt{R}_\mathrm{C}^\top\mathbf{t}_\mathrm{C} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathtt{R}_\mathrm{M}^\top & \mathtt{R}_\mathrm{M}^\top\mathbf{t}_\mathrm{M} \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{X} \\ &= \begin{bmatrix} \mathtt{R}_\mathrm{C}^\top\mathtt{R}_\mathrm{M}^\top & \mathtt{R}_\mathrm{C}^\top\mathtt{R}_\mathrm{M}^\top\mathbf{t}_\mathrm{M} & \mathtt{R}_\mathrm{C}^\top\mathbf{t}_\mathrm{C} \\ \mathbf{0}^\top & & 1 \end{bmatrix} \mathbf{X} \end{aligned} \tag{2.8}$$

where $\mathbf{X}_\mathrm{C}$ is in the camera coordinate system, $\mathbf{X}_\mathrm{M}$ is in the multi-camera coordinate system and $\mathbf{X}$ is in the world coordinate system. The forward projection function from 3D point to image point is written in homogeneous coordinate as (putting this together with Eq.(2.1))

$$\mathbf{x} = \mathtt{K}[\mathtt{I} \mid \mathbf{0}]\mathbf{X}_\mathrm{C} = \mathtt{K}\mathtt{R}_\mathrm{C}^\top \left[\mathtt{R}_\mathrm{M}^\top \mid \quad \mathtt{R}_\mathrm{M}^\top\mathbf{t}_\mathrm{M} \quad \mathbf{t}_\mathrm{C} \right] \mathbf{X}. \tag{2.9}$$

This is a general mapping given by a rigid multi-camera system with pinhole camera model. The parameters $\mathtt{R}_\mathrm{M}$ and $\mathbf{t}_\mathrm{M}$ which relate the multi-camera rig orientation and position to the world coordinate system are called the *multi-camera pose*. The parameters contained in $\mathtt{K}$ are called the *intrinsic parameters* of the camera. The transformation defined by $(\mathtt{R}_\mathrm{C}, \mathbf{t}_\mathrm{C})$ is called the *extrinsic parameters* of the multi-camera system. From the extrinsic parameters, we can deduce relative poses between cameras, called the *inter-camera poses*.

2.2 Parametrization of rotations

Parametrization of rotations is usually needed in applications such as structure-from-motion or camera calibration. We remind that the set of rotations

$$\mathbb{SO}(3) = \left\{\mathtt{R} \in \mathbb{R}^{3\times 3} | \mathtt{R}^\top \mathtt{R} = \mathtt{I}_{3\times 3}, \det(\mathtt{R}) = 1\right\} \tag{2.10}$$

forms a matrix group under the operations of matrix multiplication. In this section, we discuss several parameterizations of $\mathbb{SO}(3)$. The most classic ones are three-parameter representations (angle-axis, Euler-angles) and a four-parameter representation (quaternions).

2.2.1 Definitions

A parametrization (or a parametric representation) of $\mathbb{SO}(3)$ is a surjective and $\mathcal{C}^1$ continuous function $\mathcal{R}$ from $\mathbb{R}^k$ to $\mathbb{SO}(3)$. Since $\mathbb{SO}(3)$ is a 3D manifold (same topology as the real 3D projective space), it has 3 degrees of freedom (DoF) and the integer k is greater than or equal to 3. Let $\mathbf{r}_0 \in \mathbb{R}^k$ and $\partial\mathcal{R}(\mathbf{r}_0)$ be the Jacobian of $\mathcal{R}$ at $\mathbf{r}_0$, by considering that $\mathbb{SO}(3)$ is a subset of $\mathbb{R}^9$. In our context, we say that $\mathbf{r}_0 \in \mathbb{R}^k$ is a singularity of $\mathcal{R}$ if the rank of $\partial\mathcal{R}(\mathbf{r}_0) < 3$. If $\mathbf{r}_0$ is not a singularity, this rank is 3 and $\mathcal{R}$ is locally surjective: $\mathcal{R}$ maps a (arbitrary small) neighborhood of $\mathbf{r}_0$ to a neighborhood of rotation $\mathcal{R}(\mathbf{r}_0)$ in $\mathbb{SO}(3)$. The singularities should be avoided in the parametrization choice used in bundle adjustment [Triggs+00], [Hartley+04].

Here, we detail the case where k is minimal, i.e. $k = 3$. We have $\mathrm{rank}\partial\mathcal{R}(\mathbf{r}_0) < 3$ iff the kernel $\ker \partial\mathcal{R}(\mathbf{r}_0)$ has a non-zero dimension. For example, assume that there is a curve C (a $\mathcal{C}^1$ continuous function) from interval $]-1, 1[$ to $\mathbb{R}^3$ such that the function composition $\mathcal{R} \circ C$ is constant and $C(0) = \mathbf{r}_0$. We have $\partial\mathcal{R}(\mathbf{r}_0)\partial C(0) = \mathbf{0}$ by the Chain rule, and thus $\dim \ker \partial\mathcal{R}(\mathbf{r}_0) > 0$, which means that $\mathbf{r}_0$ is a singularity. If $\mathbf{r}_0$ is not a singularity, $\mathcal{R}$ is a locally $\mathcal{C}^1$-diffeomorphism: between a neighborhood of $\mathbf{r}_0$ and a neighborhood of $\mathcal{R}(\mathbf{r}_0)$ in $\mathbb{SO}(3)$. All parameterizations of $\mathbb{SO}(3)$ with $k = 3$ have singularities [Singla+04].

2.2.2 The angle-axis representation

It is shown in [Hartley+04] that every rotation $\mathtt{R} \in \mathbb{SO}(3)$ can be written as

$$\mathtt{R} = \exp(\Omega) = \mathtt{I} + \Omega + \Omega^2/2! + \Omega^3/3! + \cdots \tag{2.11}$$

where Ω is a 3×3 skew-symmetric matrix. The exponential map is a surjective function onto $\mathbb{SO}(3)$. A matrix Ω can be expressed in terms of the entries of a 3-vector $\mathbf{v} = (v_1, v_2, v_3)^\top$ by

$$\Omega = [\mathbf{v}]_\times = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}. \tag{2.12}$$

The vector $\mathbf{v}$ is known as the angle-axis representation of the rotation. Given a 3-vector $\mathbf{v} = \theta\hat{\mathbf{v}}$ where $\hat{\mathbf{v}}$ is a unit vector, [Hartley+04] shows that matrix $\exp[\mathbf{v}]_\times$ is precisely the rotation about the unit axis $\hat{\mathbf{v}}$ through angle θ.

The exponential map on $\exp[.]_\times : \mathbb{R}^3 \to \mathbb{SO}(3)$ can be computed using Rodrigues' formula [Hartley+04]:

$$\mathtt{R}(\mathbf{v}) = \exp[\theta\hat{\mathbf{v}}]_\times = \mathtt{I} + \sin\theta[\hat{\mathbf{v}}]_\times + (1 - \cos\theta)[\hat{\mathbf{v}}]_\times^2. \tag{2.13}$$

Note that an alternative representation $(2\pi - \theta)(-\hat{\mathbf{v}})$ represents the same rotation; thus, the angle-axis representation is not unique. We also note that $\mathtt{R}(\mathbf{v}) = \mathtt{I}$ if $||\mathbf{v}|| = 2\pi$. By using a curve on sphere of equation $||\mathbf{v}|| = 2\pi$, we see that this sphere is a set of singularities of $\mathtt{R}$. Thanks to Eq.(2.13), the concentric spheres with centre $\mathbf{0}$ and radial $2\pi\mathbb{Z}^*$ are singularities.

2.2.3 Euler angles

Euler angles describe a rotation by three successive rotation angles (α, β, γ) around the vectors of the canonical basis of $\mathbb{R}^3$. The order of the basis vector is important here. We define the following three elementary rotation matrices:

$$\mathtt{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}, \mathtt{R}_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}, \mathtt{R}_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.14}$$

then the resultant rotation matrix can be written as

$$\mathtt{R}(\alpha, \beta, \gamma) = \mathtt{R}_z(\gamma)\mathtt{R}_y(\beta)\mathtt{R}_x(\alpha). \tag{2.15}$$

In this way, it is possible to define 12 sets of Euler angles associated which 12 representations of the rotation matrix: 6 "symmetric" sets when first and last rotation occur about the same axis, and 6 "asymmetric" sets when all the three rotations occur about three distinct axes (i.e. change the ordering of $\mathtt{R}_z, \mathtt{R}_y$ and $\mathtt{R}_x$ in Eq.(2.15)). In our work, we use the representation defined by Eq.(2.15).

These matrices involve the sine and cosine of the Euler angles, and though they are nonlinear, their derivatives are easy to compute. Euler angles provide an easy to use interface to animators in form of three independent angles [Singla+04]. Given a rotation matrix $\mathtt{R}$, Appendix D.1 shows that if $\sin\beta \neq \pm 1$, the inverse transformation problem has two solutions (α, β, γ) and $(\alpha + \pi, \pi - \beta, \gamma + \pi)$. The singularities of the Euler representation form parallel and equidistant planes of equation $\beta = \pi/2 + l\pi, l \in \mathbb{Z}$ [Singla+04] (also see Appendix D.2 for a more detailed proof).

2.2.4 Quaternions

Quaternions form a group whose underlying set is the four dimensional vector space $\mathbb{R}^4$ with a multiplication operator that combines both the dot product and cross product of vectors (see Appendix E). We represent rotation matrix $\mathtt{R}$ by quaternion $\mathbf{q} = [q_r, q_i, q_j, q_k]$ as

$$\mathtt{R}(\mathbf{q}) = \frac{\mathtt{S}}{||\mathbf{q}||^2} \tag{2.16}$$

where

$$\mathtt{S} = \begin{bmatrix} q_r^2 + q_i^2 - q_j^2 - q_k^2 & 2q_iq_j - 2q_rq_k & 2q_iq_k + 2q_rq_j \\ 2q_iq_j + 2q_rq_k & q_r^2 - q_i^2 + q_j^2 - q_k^2 & 2q_jq_k - 2q_rq_i \\ 2q_iq_k - 2q_rq_j & 2q_jq_k + 2q_rq_i & q_r^2 - q_i^2 - q_j^2 + q_k^2 \end{bmatrix}. \tag{2.17}$$

This parametrization does not have singularities [Dam+98]. In practice, unit-length quaternions are often used to obtain a rotation. The set of unit-length quaternions is a sub-group whose underlying set is called $\mathbb{H}_1$. A unit quaternion $\mathbf{q}$ can be written $\mathbf{q} = (\cos(\theta/2), \sin(\theta/2)\hat{\mathbf{v}})$. Such a quaternion represents a rotation through the angle θ about the vector $\hat{\mathbf{v}}$. Note that, $\mathtt{R}(\mathbf{q}) = \mathtt{R}(-\mathbf{q})$. So both $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation.

Quaternions, however, have one extra DoF. There are four directions in which quaternion can change, but only three rotation DoF. An optimizer is free to move the quaternion off the unit quaternion sphere, leading to non-unit quaternions that hence give a redundant parameterization of rotation. Moreover, any motion in the tangent plane of sphere $\mathbb{H}_1$ will push the quaternion out of sphere. Several strategies have been developed to deal with these complications. [Ikits00] uses the fact that $||\mathbf{q}||^2 = 1$, optimizes only three components and calculates the remaining components of a quaternion with this constraint, $q_r = \sqrt{1 - q_i^2 - q_j^2 - q_k^2}$. Since it does not allow for negative q_r, this parameterization limits us in a half-part of the parameter space. [Schmidt+01] proposes a local approximation to the unit quaternion by calculating the tangent space of the unit quaternion manifold. A movement of quaternion on the unit sphere $\mathbb{S}^3$ is described by a 3-vector lying the tangential hyperplane (subspace of $\mathbb{R}^4$) of the unit sphere at the point $\mathbf{q}$. This 3-vector is represented in the local coordinate system of the tangential hyperplane and gives a direction and a distance (its length) to move on a great circle of the unit sphere. Alternatively, [Agarwal+] suggests an algebraic approach that is computationally less complex than the above approach (more details in Appendix C.2.2).

2.3 Optimization method

In this section, we briefly discuss about some algorithms in order to solve our optimization problems. Let $F : \mathbb{R}^n \to \mathbb{R}$ be $\mathcal{C}^2$ continuous function. In typical optimization problems, we find a minimizer for F that gives the minimum value of this so-called *objective function* or *cost function*:

$$\mathbf{x}^+ = \arg\min_{\mathbf{x} \in \mathbb{R}^n} F(\mathbf{x}). \tag{2.18}$$

This problem is very hard to solve in general and we are not guaranteed to find such a global minimizer $\mathbf{x}^+$ for F. Therefore, we often focus on the simpler problem of finding a

local minimizer $\mathbf{x}^*$ in a neighborhood of an initial guess $\mathbf{x}_0$ instead. Sufficient condition for a local minimizer is that the gradient of $F(\mathbf{x})$ is equal to $\mathbf{0}$ at $\mathbf{x}^*$ and the Hessian matrix of $F(\mathbf{x})$ is positive definite at $\mathbf{x}^*$.

Gradient descent. We would like to find a local minimizer in the neighborhood of $\mathbf{x}_0$. We start at $\mathbf{x}_0$ and walk iteratively $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k, \ldots$ along the direction of the negative gradient $-\mathbf{g} = -\nabla F(\mathbf{x}_k)$. Thus, we employ the following update rule:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}. \tag{2.19}$$

The factor α_k is chosen in a way we get a decrease in the value of the objective function. If no such α_k exists, a local minimizer is reached if the Hessian matrix is positive definite. The value of α_k is selected adaptively or by a line search in the downward gradient direction. A method likes this converges locally, but the convergence rate can be very slow, even in a close neighborhood of the minimum.

Newton method. Let us assume that an initial estimation $\mathbf{x}_0$ is in the neighborhood of a local minimizer $\mathbf{x}^*$ such that the Hessian matrix $\mathtt{H}$ of F is positive semi-definite in this neighborhood. Since $\mathbf{x}^*$ is a local minimizer, $\mathbf{g}(\mathbf{x}^*) = \mathbf{0}$. We can approximate $\mathbf{g}(\mathbf{x}^*)$ using the first order Taylor expansion:

$$\mathbf{g}(\mathbf{x}^*) = \mathbf{g}(\mathbf{x}_0 + \Delta) \approx \mathbf{g}(\mathbf{x}_0) + \mathtt{H}_{F(\mathbf{x}_0)}\Delta. \tag{2.20}$$

Because $\mathbf{g}(\mathbf{x}^*) = \mathbf{0}$, we get $\mathbf{x}^* = \mathbf{x}_0 - \mathtt{H}^{-1}_{F(\mathbf{x}_0)}\mathbf{g}(\mathbf{x}_0)$. This leads to the recursive update rule:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathtt{H}^{-1}_{F(\mathbf{x}_k)}\mathbf{g}(\mathbf{x}_k). \tag{2.21}$$

Let $\Delta = \mathbf{x}_{k+1} - \mathbf{x}_k$. The Newton method can be performed by repetitively solving the following linear system

$$\mathtt{H}_{F(\mathbf{x}_k)}\Delta = -\mathbf{g}(\mathbf{x}_k) \tag{2.22}$$

followed by an additive update: $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta$. In contrast to gradient descent method, this method converges especially fast in the neighborhood of the minimum. A disadvantage of this approach is that the computation of the Hessian may be difficult. The behavior of this algorithm depends strongly on the initial guess $\mathbf{x}_0$. It is possible that this procedure does not converge at all if $\mathbf{x}_0$ is far from the local minimizer $\mathbf{x}^*$.

Gauss-Newton method. Now we consider a special problem where we would like to estimate $\mathbf{x}$ by minimizing a quadratic cost, called a least squares problem:

$$F(\mathbf{x}) = \frac{1}{2}\boldsymbol{\epsilon}(\mathbf{x})^\top \Lambda \boldsymbol{\epsilon}(\mathbf{x}) \tag{2.23}$$

where $\boldsymbol{\epsilon} : \mathbb{R}^n \to \mathbb{R}^m$ is a $\mathcal{C}^2$ continuous and $\Lambda \in \mathbb{R}^{m\times m}$ is a symmetric, positive semi-definite matrix. It covers a large class of problem in different domains. Especially, the classical problem in Computer Vision is to find 3D point positions and camera parameters that minimize the reprojection error as we will see later. For high-dimensional problems, the efficient calculation of Hessian is difficult. The Gauss-Newton method is based on linear approximation of $\boldsymbol{\epsilon}$ in the neighborhood of $\mathbf{x}$: for small Δ, we write its Taylor expansion as

$$\boldsymbol{\epsilon}(\mathbf{x} + \Delta) \approx \mathbf{l}(\Delta) \equiv \boldsymbol{\epsilon}(\mathbf{x}) + \mathtt{J}(\mathbf{x})\Delta \tag{2.24}$$

where $\mathtt{J} \in \mathbb{R}^{m \times n}$ is the Jacobian matrix of $\boldsymbol{\epsilon}$. Inserting this to Eq.(2.23), we see that (with $\boldsymbol{\epsilon} = \boldsymbol{\epsilon}(\mathbf{x})$ and $\mathtt{J} = \mathtt{J}(\mathbf{x})$)

$$\begin{aligned} F(\mathbf{x}+\Delta) \approx L(\Delta) &\equiv \frac{1}{2}\mathbf{l}(\Delta)^\top \Lambda \mathbf{l}(\Delta) = \frac{1}{2}(\boldsymbol{\epsilon} + \mathtt{J}\Delta)^\top \Lambda (\boldsymbol{\epsilon} + \mathtt{J}\Delta) \\ &= \frac{1}{2}\boldsymbol{\epsilon}^\top \Lambda \boldsymbol{\epsilon} + \Delta^\top \mathtt{J}^\top \Lambda \boldsymbol{\epsilon} + \frac{1}{2}\Delta^\top \mathtt{J}^\top \Lambda \mathtt{J} \Delta \\ &= F(\mathbf{x}) + \Delta^\top \mathtt{J}^\top \Lambda \boldsymbol{\epsilon} + \frac{1}{2}\Delta^\top \mathtt{J}^\top \Lambda \mathtt{J} \Delta. \end{aligned} \tag{2.25}$$

It is easily seen that the gradient and the Hessian of L are

$$\mathbf{g}_L(\Delta) = \mathtt{J}^\top \Lambda \boldsymbol{\epsilon} + \mathtt{J}^\top \Lambda \mathtt{J} \Delta \quad \text{and} \quad \mathtt{H}_L(\Delta) = \mathtt{J}^\top \Lambda \mathtt{J}. \tag{2.26}$$

We see that $\mathbf{g}_L(\mathbf{0}) = \mathbf{g}_F(\mathbf{x})$ and $\mathtt{H}_L(\Delta)$ is independent of Δ. In the Gauss-Newton method, the linear system of Eq.(2.22) is approximated by the *normal equation*

$$\mathtt{J}^\top \Lambda \mathtt{J} \Delta = -\mathtt{J}^\top \Lambda \boldsymbol{\epsilon}. \tag{2.27}$$

This is a good approximation if $\mathbf{x}$ is close enough to a minimizer $\mathbf{x}^*$ of F.

Levenberg-Marquardt method (LM). The Newton-type methods work well close to the minimum, but elsewhere these methods cannot distinguish between local minima, saddle points and local maxima [Madsen+04]. On the contrary, the gradient descent converges globally but performs poorly close to the minimum. This leads to the Levenberg-Marquardt method [Levenberg44], [Marquardt63] which may be seen as a hybrid between Gauss-Newton and gradient descent methods. The LM algorithm is used in many software applications for solving non-linear least squares problems such as Matlab [Matlab17], MinPack [Devernay07], Eigen [Guennebaud+10], SBA [Lourakis+09], LMA [Ramadasan+17], Ceres [Agarwal+], etc. The normal equation Eq.(2.27) is altered by the *augmented normal equation* as follows:

$$(\mathtt{J}^\top \Lambda \mathtt{J} + \frac{1}{\mu}\mathtt{D})\Delta = -\mathtt{J}^\top \Lambda \boldsymbol{\epsilon}. \tag{2.28}$$

where $\mathtt{D}$ is a non-negative diagonal matrix, typically the diagonal of matrix $\mathtt{J}^\top \Lambda \mathtt{J}$. The damping parameter $\mu > 0$ has several effects. For all $1/\mu > 0$, $(\mathtt{J}^\top \Lambda \mathtt{J} + \frac{1}{\mu}\mathtt{D})$ is positive definite and this ensures that Δ is a descent direction. For large value of $1/\mu$, the update vector Δ rotates towards gradient direction, and this is good if the current state is far from the solution. If $1/\mu$ is very small, LM approaches pure Gauss-Newton and if $\mathbf{x}$ is close to the local minimizer, this method converges fast to the minimum value.

Assume that $\Lambda = \mathtt{I}$. For simplicity, we start by the basic LM algorithm [Press+96] summarized in Algorithm 1. $k_{\max}$ is the maximum number of iterations. Typically, the initial value of μ is $\mu_0 = 10^3$. If the current $1/\mu$ reduces the cost function, i.e. $F_{\text{new}} < F$, then $1/\mu$ is reduced in the next iteration to accelerate convergence. In this case, LM behaves like Gauss-Newton method. Otherwise, we increase $1/\mu$ and LM approaches gradient descent algorithm to guarantee a decreasing cost function. The iteration stops if F decreases too slowly, i.e. if $F_{\text{new}} > \alpha F$ where α is a positive parameter which is set to 0.9999 typically; and the last parameter vector is considered to be the solution.

The damping parameter μ is chosen in order to guarantee convergence of the algorithm. Various heuristic arguments put forward for the best choice for the damping parameter [Nielsen99]. A tutorial discussing non-linear least squares in general and the

Algorithm 1 Basic Levenberg-Marquardt method [Press+96]

$k \leftarrow 0$; $\mathbf{x} \leftarrow \mathbf{x}_0$
$F \leftarrow F(\mathbf{x})$; $\mathtt{A} \leftarrow \mathtt{J}(\mathbf{x})^\top \mathtt{J}(\mathbf{x})$; $\mathbf{g} \leftarrow \mathtt{J}(\mathbf{x})^\top \boldsymbol{\epsilon}(\mathbf{x})$
$stop \leftarrow$ **false**; $\mu \leftarrow \mu_0$
while $(!stop)$ **and** $(k < k_{\max})$ **do**
    $k \leftarrow k + 1$; Solve $\left(\mathtt{A} + \frac{1}{\mu}\mathtt{D}\right)\Delta = -\mathbf{g}$
    $\mathbf{x}_{\text{new}} \leftarrow \mathbf{x} + \Delta$
    $F_{\text{new}} \leftarrow F(\mathbf{x}_{\text{new}})$
    if $(F_{\text{new}} < F)$ **then**
        $\mathbf{x} \leftarrow \mathbf{x}_{\text{new}}$
        if $(F_{\text{new}} < \alpha F)$ **then**
            $\mu \leftarrow \mu * 10$
            $F \leftarrow F(\mathbf{x})$; $\mathtt{A} \leftarrow \mathtt{J}(\mathbf{x})^\top \mathtt{J}(\mathbf{x})$; $\mathbf{g} \leftarrow \mathtt{J}(\mathbf{x})^\top \boldsymbol{\epsilon}(\mathbf{x})$
        else
            $stop \leftarrow$ **true** ▷ converges
        end if
    else
        $\mu \leftarrow \mu/10$
    end if
end while

Algorithm 2 Levenberg-Marquardt method with updated μ parameter adaptively [Madsen+04, Algorithm 3.16] with a few modifications in [Agarwal+]

$k \leftarrow 0$; $\nu \leftarrow 2$; $\mathbf{x} \leftarrow \mathbf{x}_0$
$\mathtt{A} \leftarrow \mathtt{J}(\mathbf{x})^\top \mathtt{J}(\mathbf{x})$; $\mathbf{g} \leftarrow \mathtt{J}(\mathbf{x})^\top \boldsymbol{\epsilon}(\mathbf{x})$
$found \leftarrow (\|\mathbf{g}\|_\infty \le \varepsilon_1)$; $\mu \leftarrow \mu_0$
while $(!found)$ **and** $(k < k_{\max})$ **and** $(t < t_{\max})$ **and** $(\mu > \mu_{\min})$ **do**
    $k \leftarrow k + 1$; Solve $\left(\mathtt{A} + \frac{1}{\mu}\mathtt{D}\right)\Delta = -\mathbf{g}$
    $\mathbf{x}_{\text{new}} \leftarrow \mathbf{x} + \Delta$
    $\rho \leftarrow \dfrac{F(\mathbf{x}) - F(\mathbf{x}_{\text{new}})}{L(\mathbf{0}) - L(\Delta)}$
    if $(||\Delta|| \le \varepsilon_2(||\mathbf{x}|| + \varepsilon_2))$ **or** $\left(\dfrac{|F(\mathbf{x}) - F(\mathbf{x}_{\text{new}})|}{F(\mathbf{x})} \le \varepsilon_0\right)$ **then**
        $found \leftarrow$ **true**
    else
        if $\rho > \eta_1$ **then** ▷ accepted step
            $\mathbf{x} \leftarrow \mathbf{x}_{\text{new}}$; $\mathtt{A} \leftarrow \mathtt{J}(\mathbf{x})^\top \mathtt{J}(\mathbf{x})$; $\mathbf{g} \leftarrow \mathtt{J}(\mathbf{x})^\top \boldsymbol{\epsilon}(\mathbf{x})$
            $found \leftarrow (\|\mathbf{g}\|_\infty \le \varepsilon_1)$
            $\mu \leftarrow \mu / \max\left\{\frac{1}{3}, 1 - (2\rho - 1)^3\right\}$; $\nu \leftarrow 2$
        else
            $\mu \leftarrow \mu/\nu$; $\nu \leftarrow 2 * \nu$
        end if
    end if
end while

LM method in particular can be found in [Madsen+04]. In our work, we use an open-source library Ceres [Agarwal+] which includes an implementation of the LM algorithm in [Madsen+04] (see Algorithm 2). We briefly summarize it. The ratio

$$\rho = \frac{F(\mathbf{x}) - F(\mathbf{x} + \Delta)}{L(\mathbf{0}) - L(\Delta)} \tag{2.29}$$

measures the quality of the step Δ, i.e., how well the linear model $L(\Delta)$ (Eq.(2.25)) predicts the decrease of the non-linear objective $F(\mathbf{x} + \Delta)$. The idea is to increase or to decrease the radius of the search region of step Δ depending on how well the linearization L predicts the behavior of the non-linear objective F, which in turn is reflected in the value of ρ. By construction, the denominator in Eq.(2.29) is positive. Furthermore, the numerator is negative if the step is not downhill, i.e. the step Δ is too large and should be reduced. A small value of ρ indicates that we should increase the damping parameter $1/\mu$ and thereby increase the penalty on large step Δ. Otherwise, a large value of ρ indicates that $L(\Delta)$ is a good approximation of $F(\mathbf{x}+\Delta)$ for the computed Δ and the damping parameter $1/\mu$ may be deduced. Notice that if a damping parameter is accepted, we check that ρ is positive and greater than a lower threshold for relative decrease η_1. Typically, $\eta_1 = 1e-3$ and the initial $\mu_0 = 1e4$.

There are several *stopping criteria* for the LM algorithm. We consider that the algorithm converges if

- Function tolerance:
$$\frac{|F(\mathbf{x}) - F(\mathbf{x}_{\text{new}})|}{F(\mathbf{x})} \leq \varepsilon_0, \tag{2.30}$$
where, $|F(\mathbf{x}) - F(\mathbf{x}_{\text{new}})|$ is the change in objective function value (up or down) in the current iteration of LM.

- Gradient tolerance:
$$||\mathbf{g}||_\infty \leq \varepsilon_1, \tag{2.31}$$
where ε_1 is a small, positive number (reminder: $\mathbf{g}(\mathbf{x}^*) = \mathbf{0}$ if $\mathbf{x}^*$ is a local minimizer).

- Parameter tolerance:
$$||\Delta|| \leq \varepsilon_2(||\mathbf{x}|| + \varepsilon_2). \tag{2.32}$$
when the change in $\mathbf{x}$ is small.

The two first criteria can be achieved when $\mathbf{x}$ is very close to a local minimizer $\mathbf{x}^*$. The last criterion come into effect if very small values of ε_0 and ε_1 are chosen. It arises when there is a poor prediction of the linear model $L(\Delta)$ (Eq.(2.25)). In this case, the damping parameter $1/\mu$ will be augmented in every step. When $1/\mu$ grows fast, it results in small Δ, and the process should be stopped by Eq.(2.32). As in all iterative processes, we need a safeguard against an infinite loop:

- A maximum number of iterations and a maximum amount of time for which the solver should run:
$$k \geq k_{\max} \quad \text{and} \quad t \geq t_{\max}. \tag{2.33}$$

- The solver can also terminate if the damping parameter becomes smaller than $\mu_{\min}$

$$\mu \leq \mu_{\min}. \tag{2.34}$$

In our work, we set $\varepsilon_0 = 1e-08$, $\varepsilon_1 = 1e-10$, $\varepsilon_2 = 1e-16$, $k_{\max} = 200$ and use Ceres's default: $t_{\max} = 1e6$ (seconds), $\mu_{\min} = 1e-32$.

2.4 Reconstruction from image sequences

In this section, we summarize the *Structure from Motion* or SfM algorithm [Mouragnon+07], [Mouragnon+09] that we use. The objective is to automatically compute a reconstruction (both camera poses and 3D scene points) from a video sequence captured by a central calibrated (multi-)camera. The SfM can be split in three distinct tasks:

i) track feature points throughout the sequence;

ii) estimate an initial reconstruction which is used as a starting point for

iii) bundle adjustment.

2.4.1 Corresponding features

A video sequence has several advantages over an arbitrary set of images: there is an ordering on the images; the baseline (distance) between two successive frames is small. Thus the feature matches are obtained and assessed more easily. In addition, some of our methods need a video sequence, e.g. bundle adjustment with a model of continuous-time camera trajectory.

Interest point detection and matching. For each frame, Harris corners [Harris+88] are detected and matched which ones from the previous frame. For each interest point in the previous frame, we define in the new frame a region of interest whose center is this position. We select possible corresponding points in the new frame which are inside the search zone. Then a *Zero mean Normalized Cross-Correlation* (ZNCC) score is computed between the point in the previous frame and all the potential candidates in the new frame. The ZNCC score is defined as follows

$$\mathrm{ZNCC}(\mathbf{p}_1, \mathbf{p}_2) = \frac{\sum_{\mathbf{d}\in\nu_w} (L_1(\mathbf{p}_1+\mathbf{d}) - \bar{L}_1(\mathbf{p}_1))(L_2(\mathbf{p}_2+\mathbf{d}) - \bar{L}_2(\mathbf{p}_2))}{\sqrt{\sum_{\mathbf{d}\in\nu_w} (L_1(\mathbf{p}_1+\mathbf{d}) - \bar{L}_1(\mathbf{p}_1))^2}\sqrt{\sum_{\mathbf{d}\in\nu_w} (L_2(\mathbf{p}_2+\mathbf{d}) - \bar{L}_2(\mathbf{p}_2))^2}} \tag{2.35}$$

where

$$\bar{L}_i(\mathbf{p}_i) = \frac{1}{w^2} \sum_{\mathbf{d}\in\nu_w} L_i(\mathbf{p}_i+\mathbf{d}), \tag{2.36}$$

$\mathbf{p}_1$ and $\mathbf{p}_2$ are the two compared points, $L_i(\mathbf{p})$ is the luminance value at the point $\mathbf{p}$ and ν_w is the $w \times w$ neighborhood of $\mathbf{p}$. The best candidate with highest score is selected and put to a list of corresponding point pairs between these two frames.

Tracking and key frame sampling. The disadvantage of a small baseline between two frames is that the 3D structure is estimated poorly. It may be mitigated by tracking through the sequence so that the baseline is enough large. If the camera motion between two frames is large enough, the computation of the epipolar geometry is a well conditioned problem. That leads to the concept of *key frame*. From a video sequence, it is necessary to extract key frames being not too close and not too far to each other while still being able to find enough point correspondences between the key frames. Among algorithms proposed to do that, the method in [Royer+07] is simple and works well. It is detailed as follows with an improvement in the case of slow camera motion [Litvinov15]:

1. The first frame is always selected as the first key frame.
2. The current frame is rejected if the image motion between it and previous key frame is small, i.e. if 70% of its matches have 2D motion less than D (a user defined threshold in pixels).
3. The current frame is the second keyframe if it is not rejected by 2D motion condition above and it has at least N_2 matches with the first key frame.
4. When key frames $I_0, I_1, \ldots, I_i$ $(i \geq 1)$ are selected, a non-rejected previous frame is chosen as key frame I_{i+1} if
   i) it has at least N_2 matches with the key frame I_i and
   ii) is has at least N_3 matches with the key frame I_{i-1}
   iii) the current frame does not meet i) or ii).

If the previous frame is not a key frame, then we suppress all its observations from the currently maintained tracks and replace them by the matches of the current frame. In this way, the points are continuously tracked between the current key frame and the future key frame.

2.4.2 Initial reconstruction

Epipolar geometry and pose estimation. In this stage, we need to estimate a camera (rig) pose defined by a pair $(\mathtt{R}, \mathbf{t})$ in the world coordinate system. We first assume that the cameras are (approximately) calibrated and the scene knowledge is not available (no 3D point position is known). The *epipolar geometry* describes the relative position and internal parameters of two views and does not depend on the scene structure. This constraint is represented by a 3×3 matrix called the *fundamental matrix* [Hartley+04]. Since the cameras are calibrated, we compute the *essential matrix* [Faugeras93] instead of the fundamental matrix. We explain it in terms of the rays in space corresponding to two image points.

Assume that 3D point $\mathbf{X}$ is observed by two distinct view points such that the corresponding ray direction are $\mathbf{d}_1$, $\mathbf{d}_2$ in the coordinate systems of these frames. The essential matrix $\mathtt{E}$ meets the condition

$$\mathbf{d}_2^\top \mathtt{E} \mathbf{d}_1 = 0. \tag{2.37}$$

Let $(\mathtt{R}_1, \mathbf{t}_1)$ be unknown pose of the first frame and $(\mathtt{R}_2, \mathbf{t}_2)$ be unknown pose of second frame in the world coordinate system. So, $\mathtt{R}_1\mathbf{d}_1$ and $\mathtt{R}_2\mathbf{d}_2$ are the directions of $\mathbf{d}_1$ and $\mathbf{d}_2$,

respectively in the world coordinate system. The vectors $\mathbf{t}_1$, $\mathbf{t}_2$, $\mathbf{t}_1 + \mathtt{R}_1\mathbf{d}_1$ and $\mathbf{t}_2 + \mathtt{R}_2\mathbf{d}_2$ are coplanar. In other words, $\mathbf{0}$, $\mathbf{t}_2 - \mathbf{t}_1$, $\mathtt{R}_1\mathbf{d}_1$ and $\mathbf{t}_2 - \mathbf{t}_1 + \mathtt{R}_2\mathbf{d}_2$ are lying on the same plane defined by $d + \mathbf{n}^\top\mathbf{X} = 0$ where $\mathbf{n}, \mathbf{X} \in \mathbb{R}^3$, $\mathbf{n} \neq \mathbf{0}$ and $d \in \mathbb{R}$. Since this plane passes through $\mathbf{0}$, $d = 0$. Furthermore, $\mathbf{n} = (\mathbf{t}_2 - \mathbf{t}_1) \times \mathtt{R}_1\mathbf{d}_1 = [\mathbf{t}_2 - \mathbf{t}_1]_\times \mathtt{R}_1\mathbf{d}_1$ where $\times$ is cross product between two vectors and $[\mathbf{v}]_\times$ is the skew-symmetric 3×3 matrix of a vector $\mathbf{v}$, defined in Eq.(2.12). Note that $\mathbf{n}$ and $\mathbf{t}_2 - \mathbf{t}_1 + \mathtt{R}_2\mathbf{d}_2$ are perpendicular (since $d = 0$), hence

$$(\mathbf{t}_2 - \mathbf{t}_1 + \mathtt{R}_2\mathbf{d}_2)^\top [\mathbf{t}_2 - \mathbf{t}_1]_\times \mathtt{R}_1\mathbf{d}_1 = 0. \tag{2.38}$$

Because $(\mathbf{t}_2 - \mathbf{t}_1)^\top \mathbf{n} = 0$, we obtain

$$\mathbf{d}_2^\top \mathtt{R}_2^\top [\mathbf{t}_2 - \mathbf{t}_1]_\times \mathtt{R}_1\mathbf{d}_1 = 0. \tag{2.39}$$

So we define the essential matrix $\mathtt{E}$ in Eq.(2.37) as following

$$\mathtt{E} = \mathtt{R}_2^\top [\mathbf{t}_2 - \mathbf{t}_1]_\times \mathtt{R}_1. \tag{2.40}$$

The pose of the first frame can be defined at the origin of the world coordinate system with identity matrix orientation. The essential matrix, $\mathtt{E} = \mathtt{R}_2^\top [\mathbf{t}_2]_\times$ is a homogeneous quantity and has only five DoF: 3 for $\mathtt{R}_2$ and 3 for $\mathbf{t}_2$ but the scale is not significant. If there are sufficiently many point correspondences (at least 5 points [Nistér04]) to compute the essential matrix up to scale then a pair of camera poses corresponding to the essential matrix is computed using the relation Eq.(2.40) by a singular value decomposition of $\mathtt{E}$. An alternative and simple method estimates the essential matrix linearly using 8-point correspondences or using 7-point correspondences in [Hartley+04].

Now we assume that the pose $(\mathtt{R}_1, \mathbf{t}_1)$ is known, $(\mathtt{R}_2, \mathbf{t}_2)$ is unknown and some 3D point coordinates $\mathbf{X}$ observed by two frames and their set of projected points $\mathbf{p} \in \mathbb{R}^2$ in the corresponding image have already been computed. According to Section 2.1, $\mathbf{p} = f(\mathbf{I}, \mathtt{R}_2, \mathbf{t}_2, \mathbf{X})$ where f is the projection function and $\mathbf{I}$ is the camera parameters (both intrinsic and extrinsic parameters). Since the cameras are calibrated, each 3D point gives rise to 2 equations in unknown parameters of $(\mathtt{R}_2, \mathbf{t}_2)$. The camera pose has 6 DoF, so only three 3D points is necessary. We use Grunert's method [Haralick+94] to compute the pose.

Triangulation. When a pair of consistent camera matrices is reconstructed, we wish to compute the initial estimation of 3D points. The rays back-projected from the image points are inaccurate due to noise. This means that these rays never intersect in 3D space. A simple triangulation method called *middle point* algorithm in [Faugeras93] is to find the point $\mathbf{X}$ which is exactly at the middle of the shortest line segment that joins the two projection rays $(\mathbf{d}_1, \mathbf{d}_2)$. The estimated point does not exactly satisfy the projection relations and is not an optimal solution. Thus it should be refined.

The best solution requires the definition and minimization of a cost function. The approaches in [Hartley+04] consist of minimizing a sum of squares reprojection error that depends on the camera model. [Mouragnon+09] defines an angular error $\boldsymbol{\epsilon}$ as the angle between two rays. In practice, $\boldsymbol{\epsilon}$ is a 2D vector whose Euclidean norm is equal to the tangent of the angle between the given back-projected ray $\mathbf{d}$ and the direction $\mathbf{X} - \mathbf{0}$ of the line defined by the camera center $\mathbf{0}$ and 3D point $\mathbf{X}$. Let $\mathtt{R}_\mathbf{d}$ be a rotation such that $\mathtt{R}_\mathbf{d}\mathbf{d} = \mathbf{k}$ where $\mathbf{k} = (0, 0, 1)^\top$. Let function $\pi((X, Y, Z)^\top) = (X/Z, Y/Z)^\top$ where

$(X, Y, Z) \in \mathbb{R}^3$. We seek to minimize the modulus of

$$\boldsymbol{\epsilon} = \pi(\mathtt{R}_{\mathbf{d}}(\mathbf{X} - \mathbf{0})). \tag{2.41}$$

Using $(X, Y, Z)^\top = \mathtt{R}_{\mathbf{d}}(\mathbf{X} - \mathbf{0})$, we have $||\boldsymbol{\epsilon}||^2 = (X^2 + Y^2)/Z^2 = \tan^2(\mathbf{k}, \mathtt{R}_{\mathbf{d}}(\mathbf{X} - \mathbf{0})) = \tan^2(\mathbf{d}, \mathbf{X} - \mathbf{0})$. It is a suitable approximation of the squared angle if it is small. In the ideal case, $\mathbf{d}$ and $\mathbf{X} - \mathbf{0}$ are parallel, we have $||\boldsymbol{\epsilon}|| = 0$ which is equivalent to an image reprojection of zero pixel.

Principle of robust estimation. Until now, we calculate the camera poses and 3D points assuming good matching. Unfortunately, in practice, the *outliers* due to noise in measured image or false matches, will severely corrupt the precision of computation. In the following, we give a description of a general and very successful robust estimator - the *RANdom SAmple Consensus* (RANSAC) [Fischler+81] which allows to detect the outliers in the correspondences. More generally, we wish to fit a model and the random sample consists of a subset of m elements of the data that is sufficient to determine the model. The score for this model is measured by the number of *inliers* that lies within a distance threshold. The random procedure is repeated a number of iterations M and we retain the estimation which has the largest number of inliers. How do we determine the number of required iterations M? A sample is accepted if it consists of m good observations. If the whole set of observations contains up to a fraction ϵ of outliers, then the probability that at least one of M subsamples is accepted is given by

$$P = 1 - [1 - (1 - \epsilon)^m]^M. \tag{2.42}$$

So the number of iterations for given value of m and ϵ meets

$$M \geq \frac{\log(1 - P)}{\log(1 - (1 - \epsilon)^m)}. \tag{2.43}$$

P must be near to 1, typically $P = 0.99$.

Robust estimation for triangulation. In our work, we wish to estimate a 3D point $\mathbf{X}$ with RANSAC. Let $T_{\mathbf{X}}$ be the track of N_{obs} observations of $\mathbf{X}$ in image. We compute a 3D point $\mathbf{X}$ from two randomly selected correspondences. Then, we compute the value of angular error in Eq.(2.41) for each observation and check whether the observation fits the model, i.e. the angular error $||\boldsymbol{\epsilon}_i||^2 < e_{\max}$ where $e_{\max}$ is a user defined threshold and $\boldsymbol{\epsilon}_i = \pi(\mathtt{R}_{\mathbf{d}_i}(\mathbf{X} - \mathbf{0}))$ as Eq.(2.41). The number of inliers in the track is determined and the estimate $\mathbf{X}$ is retained if the number of inliers is the largest.

Robust estimation for three first key frames I_0, I_1 and I_2. We compute the poses for the three first key frames when the 3D scene knowledge is not yet available. Let I_0, I_1 and I_2 be the three first key frames. The RANSAC algorithm is applied to the track set in order to estimate the first triple pose. The sample size is five, since five tracks (correspondences) determine an essential matrix. The frame coordinate system associated to I_0 is taken as the world coordinate system. The essential matrix between I_0 and I_2 is estimated using the 5-point algorithm [Nistér04]. RANSAC retains the estimation with the largest number of inliers. Then the pose of key frame I_2 is computed. The 3D coordinates of the points associated to all the tracks are computed using the correspondences from I_0 and I_2 in the manner as described in the previous paragraph. For the key frame I_1, its pose is estimated using Grunert's algorithm [Haralick+94] with three 3D points. The final pose is chosen after RANSAC for which the number of inliers is the largest.

Robust estimation for key frame I_i ($i \geq 3$). For the newly selected key frame I_i, using Grunert's method [Haralick+94], its pose is robustly estimated from the 3D points set that is computed from key frames $I_0, I_1, \ldots, I_{i-1}$ and matched to image points in the key frame I_i. Then, new 3D points (i.e those only observed in the 3 last key frames) are reconstructed using the middle point triangulation method and RANSAC process. Outliers are eliminated by RANSAC and the results are refined from inliers using local bundle adjustment that will be mentioned in the next subsection.

2.4.3 Bundle adjustment

Assume that a set of 3D points $\mathcal{X} = \{\mathbf{X}_j\}$ is observed by a set of key frames (view points) $\mathcal{I} = \{(\mathtt{R}_k, \mathbf{t}_k)\}$. Let $\mathbf{p}_j^k$ be the observation of the j-th 3D point $\mathbf{X}_j$ as seen by the k-th camera I_k and $\mathcal{P}_{\text{obs}} = \{\mathbf{p}_j^k\}$ be the set of all inlier observations. Due to the noise in image measurement, the estimation using RANSAC in the previous subsection is not perfect and the reprojection errors are not zero. We wish to estimate the frame poses and 3D points that minimize the cost function F of the observations for every view where the 3D points appear as inliers, i.e.

$$F = \sum_{\mathbf{p}_j^k \in \mathcal{P}_{\text{obs}}} ||\boldsymbol{\epsilon}_j^k||^2 \tag{2.44}$$

where $\boldsymbol{\epsilon}_j^k$ (involving the back-projected ray of the observation $\mathbf{p}_j^k$) is defined in Eq.(2.41).

The estimation of camera poses and 3D points minimizing such a cost function is known as *bundle adjustment* (BA) [Triggs+00]. This method is a non linear optimization problem and is numerically solved using a LM method in Section 2.3. This problem is a large sparse parameter estimation problem due to the large number of estimated parameters. The special sparsity structure is exploited to solve the problem much more efficiently. Furthermore, it requires a good initialization. It is generally used as a final step of any reconstruction algorithm. Here, we use the results estimated by RANSAC as initialization for BAs.

Incremental structure from motion. The computation of geometry presented in Subsection 2.4.2 does not give a good solution. Let I_i be the last key frame. The computation of the key frame I_i depends on the previous key frames and the error can systematically accumulate over the sequence. It is not a good idea to compute all the camera locations and to use the BA only once to the whole sequence. Moreover, it possible to use a global BA to the keyframes $I_0, I_1, \ldots, I_i$ after every added keyframe I_i, but this is very inefficient for large sequence. In order to solve this problem, a better solution is to use local BA. The idea is to reduce the number of parameters involved in the optimization process after every added keyframe I_i. Only the pose parameters of the n last key frames $\mathcal{I}_i$ and the chosen 3D points in $\mathcal{X}_i$ accounting for the 2D reprojections in the N ($N \geq n$) last key frames are optimized. Thus, $\mathcal{I}_i = \{I_{i-n-1}, \ldots, I_i\}$ and $\mathcal{X}_i$ contains all the 3D points observed as inliers in a key frame in $\mathcal{I}_i$. The cost function F_i is the sum of the angle errors of points $\mathbf{X}_j \in \mathcal{X}_i$ in the N last key frames as follows

$$F_i(\mathcal{I}_i, \mathcal{X}_i) = \sum_{I_k \in \{I_{i-N-1}, \ldots, I_i\}} \sum_{\mathbf{X}_j \in \mathcal{X}_i} ||\boldsymbol{\epsilon}_j^k||^2. \tag{2.45}$$

We choose typically $n = 3$ and $N = 7$.

In practice, the optimization process takes place in two LMs. The first LM is performed using the inliers classified by RANSAC, then the inliers/outliers of all observations are updated. The result is re-estimated using the new inliers. Moreover, we note that all the poses before I_i have already optimized at preceding stages, thus the estimation of I_i is close to an optimal solution. Thus the number of necessary iterations for each stage is quite low and the process converges rapidly (with 5 iterations or less in our case). In this manner, 3D structure scene and poses are computed for large sequences. We should note that the cameras are calibrated, in other words, the intrinsic parameters are not refined in this optimization process.

2.5 Conclusion

This chapter presented preliminaries needed in our work. We described the projection model for omnidirectional multi-cameras. We wrote the equations of projection from 3D to image for the basic pinhole model. We also reminded different representations of rotation of the 3-dimensional Euclidean space. We described the SfM algorithm used to reconstruct the camera position and 3D structure scene from an input video. Of course, this algorithm is not the only option to compute SfM. But this one is widely used at Institute Pascal and so its performances and limitations are well known to us. This makes it a good choice. The results of reconstruction by SfM are used as a starting point for our work: synchronization and self-calibration.

Chapter 3

State of the art

There are a number of publications dealing with design, theory and applications of omnidirectional multi-cameras. Only the works that introduce important concepts close enough to the subject of this dissertation are mentioned in this chapter. The overview of the state-of-the-art is divided to six steps. We begin with works on estimation of intrinsic parameters in Section 3.1. Since we wish to use multiple cameras as one sensor, the cameras must be synchronized. We continue with the synchronization step for two or more video sequences in Section 3.2. This step is needed for the precision of the estimation of extrinsic parameters in the next Section 3.3. In Section 3.4, the brief historical perspective and state-of-the-art of monocular and multi-camera rolling shutter is found. We also introduce works in the context of a general multi-sensor in Section 3.5. Finally, we finish with works on 3D reconstructions from omnidirectional multi-cameras (Section 3.6).

Contents

3.1 Monocular calibration

Camera calibration process has important research and application value in Computer Vision. Its precision directly affects the quality of 3D reconstruction. If the camera is calibrated (intrinsic parameters only) and an image point is known, the corresponding ray in the camera coordinate system is uniquely determined. The calibration procedure allows us to set numeric value in the calibration matrix or the projection matrix. Although the monocular (self-)calibration is not the topic of the thesis, we briefly review this topic.

In Section 2.1, we introduced the basic pinhole model for lenses that perform ideal central projection: all lines in 3D are projected to lines in the images. This is not the case with the real lens such as wide-angle lens, (para, hyperbolic, spherical) catadioptric cameras, fish-eyes, etc. Such a typical lens performs distortion of several pixels: lines in 3D are projected to curves in the images. There are a number of camera models and calibration approaches that are proposed in the literature. We reminded radial distortion that is defined by distortion function that maps radial distance (distance between image point and distortion center) in the distorted image to either radial distance in the undistorted image or the incidence angle between camera ray and principal ray. We also mentioned the equiangular case where the incidence angle between camera ray and principal ray is proportional to radial distance in the distorted image. According to existing algorithms, camera calibration methods can be classified into two categories: photogrammetric calibration uses reference objects with known geometry and self-calibration makes no or a few assumptions about the particular structure of the scene being viewed.

Photogrammetric calibration takes the advantage of a given calibration reference object with known shape and size. The calibration of one camera from the known scene is typically a two stage process. The projection matrix is estimated from the coordinate points of the known scene. The camera pose and intrinsic parameters are estimated from the projection matrix using matrix factorization. Figure 3.1 demonstrates some calibration patterns used widely. There are many implementations of camera calibration with 2D or 3D pattern: [Tsai86], [Lavest+98], [Zhang00], [Sturm+11], [Bradski00] OpenCV library, [Bouguet02] toolbox Matlab^{TM}.

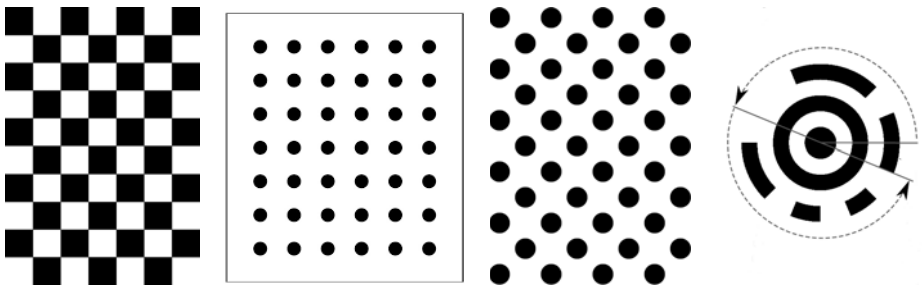


FIGURE 3.1: Calibration patterns: chessboard, symmetric circle board, asymmetric circle board, circular code.

A survey of self-calibration method for perspective cameras before 2003 can be found in [Hemayed03], [Hartley+04]. The camera self-calibration methods do not use a calibration pattern and attempt to calibrate the camera by finding intrinsic parameters that are consistent with the underlying projective geometry of an image sequence. Some self-calibration methods directly solve Kruppa equation from image pairs (which

is based on the correspondence of epipolar lines tangent to a conic). Other methods involve the use of the absolute dual quadric (which can be considered as calibration object) [Hartley+04, Chapter 19] and its projection (the dual image of the absolute conic) over many images. This approach is summarized in three steps: projective reconstruction from the given images, self-calibration assuming that pixels are squares, and refinement using BA [Triggs+00]. Some works extend the self-calibration techniques to use other constraints: camera motion or scene constraints. The camera motion constraints consider restricted motions: pure translation, pure rotation, planar motion, etc. Moreover, some methods propose to use scene constraints. For example, the *plumb line* method uses straight lines in the scene to provide constraints on intrinsic parameter: [Brito+13] for radial distortion model, [Zhang+15] for fish-eye cameras. However, straight lines are not always available in the scene, e.g. in outdoor environments. And when they are present, it is not easy to detect. Therefore, such methods often require supervision to ensure that curves in the real scene are not confounded with distorted lines. [Sturm+11] gives an overview of the vast number of camera models in the literature and the results on epipolar and multi-view geometry as well as various calibration and self-calibration approaches for different camera models.

In the context of our work, we have only videos taken in a rigid environment using an omnidirectional multi-camera system and we do not make special assumption about the camera motion. Some interesting approaches proposed in [Fitzgibbon01], [Micusik+06] require nothing more than images and the rigid scene assumption which allow to estimate fundamental matrices. The method in [Fitzgibbon01] is used only for cameras with a standard angle of view (less than 180^o), that are modeled by one radial distortion parameter. The simultaneous estimation of a single lens distortion coefficient and the fundamental matrix is expressed as a Quadratic Eigenvalue Problem. [Micusik+06] extends the work in [Fitzgibbon01] and proposes a method for fish-eye lens and catadioptric system. The authors also show that the fundamental matrix can be estimated from a small number of correspondences by solving a Polynomial Eigenvalue Problem (PEP). The first remark about these works is that the principal point (the center of radial symmetry) has to be known or is assumed to be in the center of image, and the camera has to have square pixels. With these assumptions, in [Micusik+06] using equiangular approximation, the circular (approximately) known FoV gives an initial estimate which is not precise but accurate enough to get a working PEP. The second remark is that the epipolar geometry can be estimated from a small number of correspondences using a method relying on the solution of generalized eigenproblem, whose convergence is well studied, and for which fast algorithms exist. The technique can be integrated into a robust estimation approach (RANSAC) to reject mismatches.

Degenerate configurations

The term *degenerate configuration* is used to denote the configurations of cameras and point correspondences where we are not able to self-calibrate the camera or to build a 3D reconstruction uniquely. 3D reconstruction and self-calibration are well known to have ambiguous results for some configurations of camera motion and scene structure, referred to as the *critical motion* and the *critical surface*, respectively. In this subsection, we summarize some degenerate configurations reported in the literature.

For perspective camera model (distortion-free case), degenerate configurations have been extensively studied, e.g. for Euclidean reconstruction with calibrated camera [Kahl+99], [Kahl+02] and self-calibration [Sturm97]. A survey of critical motion for

monocular perspective self-calibration has been presented by [Sturm97]. For radial distortion self-calibration (including our case), [Micusik+06], [Brito+13] report a special degenerate case - forward motion. Suppose that the principal axis $\mathbf{k}$ lies along the Z-axis and a motion of camera is along the principal axis. There exists an ambiguity in the estimated parameter of the camera model. Moreover, [Micusik+06] also shows a special case in which pure translation is ambiguous if all point displacements are parallel to camera translation in XY-plane. A full analysis of the degenerate configuration for radial distortion self-calibration under a general radial distortion model is presented in [Wu14]. According to the author, the method can be applied to both standard angle cameras and central omnidirectional cameras (for example, fisheye). Assume that the camera moves with an instantaneous translational velocity $\mathbf{t}$ and an instantaneous rotational velocity $\boldsymbol{\omega}$. Using motion field approach, the author solves for critical surface pair that can lead to the same 2D motion field under different radial distortion and possibly different camera motions. In general, critical surfaces are complicated and depend on radial distortion function and camera motion. The author mentions an example where unmanned aerial vehicles (UAVs), e.g. cameras pointing to ground, are used to capture images. The camera motions are parallel to the ground. The visible surface is near-planar and thus the cameras are not completely calibrated. Moreover, there exist critical motions under which self-calibration algorithm can fail and any surface is ambiguous for estimating radial distortion. [Wu14] finds the following critical motions

$$\mathbf{t} \times \mathbf{k} = \boldsymbol{\omega} \times \mathbf{k} = \mathbf{0} \tag{3.1}$$

where $\times$ is vector cross product. Note that the critical motion involves both translation along the principal axis and rotation around the principal axis. The forward motion degeneracy reported in [Micusik+06], [Brito+13] is special case when $\boldsymbol{\omega} = \mathbf{0}$. These critical motions for radial distortion case is a subset of that for distortion-free case. In practice, these degenerate configurations should be avoided in real capture for our wide-angle cameras.

3.2 Synchronization

The synchronization is the estimation of the temporal relationship between two or more video sequences. Synchronizing multiple video sequences can be approached in a pairwise manner. We assume that each video has a constant frame rate. Let t, t' denote the frame indices from the first and second video sequences, respectively, recorded at the same instant. We have

$$t' = \alpha t + \Delta t \tag{3.2}$$

where α is the ratio of frame rates of the two sequences and Δt is an initial temporal offset between the sequences. This section describes existing methods to estimate Δt and α for video sequences acquired by static or jointly moving cameras as well as independently moving cameras.

3.2.1 Stationary or jointly moving cameras

Several solutions to the problem of video synchronization can be found in the literature. The distinction is based on the considered assumptions. Methods assume

that the cameras are static or rigidly hold together (they move while recording), and that there exist correspondences between the features observed in the two videos.

Caspi et al. publishes a series of papers addressing this problem [Caspi+00], [Caspi+02a], [Caspi+02b], [Caspi+06]. It can be summarized as follows. For each pixel (u, v) at frame (time) t in one sequence, finds its corresponding time t' and position (u', v') in other sequence. Let $\mathbf{x}(t)$ be a *space-time point* in the reference sequence S (namely, a homogeneous pixel $(u, v, 1)$ at frame (time) t), and let $\mathbf{x}'(t')$ be the matched space-time point in sequence S'. The first method proposed in [Caspi+00], [Caspi+02b] is called "direct intensity-based sequence alignment" method. Two cameras have the same center. It requires neither detection nor tracking of moving objects. It is also called pixel-based method, i.e. featureless synchronization method using homography matrices. It can handle much more complex scene dynamics, such as changes in scene illumination and non-rigid object motion. A dimming and a brightening of a light source can provide sufficient information to align two sequences. Since global changes in illumination produce prominent temporal derivative, even homogeneous image regions contribute temporal constraints to the direct sequence-to-sequence alignment. Let $L(\mathbf{x}(t))$ be the luminance value of sequence S at the space-time point $\mathbf{x}(t)$ and $L'(\mathbf{x}'(t'))$ be the luminance value of sequence S' at the space-time point $\mathbf{x}'(t')$. Exploiting spatio-temporal brightness variations within each sequence, the authors recover the spatio-temporal displacement parameters by minimizing squared luminance distances between corresponding pixels in synchronous frames

$$\arg\min_{\mathtt{H},\Delta t} \sum_{\mathbf{x}(t)} ||L(\mathbf{x}(t)) - L'(\mathbf{x}'(t'))||^2 \tag{3.3}$$

where t' is defined as Eq.(3.2) with known α, and $\mathbf{x}' = \mathtt{H}\mathbf{x}$ with a homography $\mathtt{H}$.

Moreover, the second method presented in [Caspi+02b] (also published in the longer version [Caspi+06]), is called feature-based sequence alignment and exploits dynamic changes which are due to moving object/moving points. Therefore, this method requires detection and tracking of such object. Two sequences are related by a homography $\mathtt{H}$ (same center case) or by a fundamental matrix $\mathtt{F}$ (different center case). The aim of this approach is to minimize the following error function Eq.(3.4) in the homography case or Eq.(3.5) in the fundamental matrix case

$$\arg\min_{\mathtt{H},\Delta t} \sum_{t} ||\mathbf{x}'(\alpha t + \Delta t) - \mathtt{H}\mathbf{x}(t)||^2, \tag{3.4}$$

$$\arg\min_{\mathtt{F},\Delta t} \sum_{t} ||(\mathbf{x}'(\alpha t + \Delta t))^\top \mathtt{F}\mathbf{x}(t)||^2 \tag{3.5}$$

where the ratio α is known. Since t' in Eq.(3.2) is not necessarily an integer value, i.e. allowing a sub-frame time offset, the value of $\mathbf{x}'$ at t' is interpolated from the adjacent integer time: $t_1 = \lfloor t' \rfloor$ and $t_2 = \lceil t' \rceil$. Using the results of Eqs.(3.4) and (3.5) and then fixing $\mathtt{P}$ (denotes either homography transformation $\mathtt{H}$ or fundamental matrix $\mathtt{F}$), the authors refine Δt. The sub-frame time offset accuracy is obtained by searching for $\beta = t' - t_1$ $(1 \geq \beta \geq 0)$ that minimizes the following term

$$\arg\min_{\beta} \sum_{t} d\left(\mathbf{x}(t), (1-\beta)\mathbf{x}'(t_1) + \beta\mathbf{x}'(t_2)\right) \tag{3.6}$$

where $d(.)$ is either homography distance or fundamental distance. The closely approach proposed in [Carceroni+04] relies on the case of an arbitrary number of stationary cameras. Using epipolar geometry and RANSAC, moving scene points are tracked in each video sequence and their locations in time between frames are approximated by a linear interpolation. The similar work in [Wedge+06] is also based on epipolar geometry; however, the object motion is exploited to converge to the correct solution. The authors show that their method can reach to sub-frame accuracy, and the influence of image noise on the image point coordinates in the trajectory is not significant.

[Tresadern+03], [Rao+03] present a strategy based on rank constraint. Assuming two sequences and affine projection, [Tresadern+03] addresses non-rigid objects using measurement matrix $\mathtt{M}$ - a $4 \times N$ matrix of N image points in two corresponding frames. The measurement matrix $\mathtt{M}$ is decomposed using SVD

$$\mathtt{M} = \begin{bmatrix} u_1^1 & \cdots & u_1^n & \cdots & u_1^N \\ v_1^1 & \cdots & v_1^n & \cdots & v_1^N \\ u_2^1 & \cdots & u_2^n & \cdots & u_2^N \\ v_2^1 & \cdots & v_2^n & \cdots & v_2^N \end{bmatrix} = \begin{bmatrix} \mathtt{P}_1 \\ \mathtt{P}_2 \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 & \cdots & \mathbf{X}_n & \cdots & \mathbf{X}_N \end{bmatrix} = \mathtt{P}_{4\times 3}\mathtt{X}_{3\times N} \tag{3.7}$$

where $(u_i^n, v_i^n)^\top$ is the vector of image coordinates of the n-th feature in the i-th view, $\mathtt{P}$ is an affine projection matrix and $\mathtt{X}$ consists of 3×1 vectors of coordinates in 3D space of the n-th feature. In the ideal condition, i.e. synchronized and non-noisy case, $\text{rank}\mathtt{M} = 3$. But in general case with noise and matching error, $\text{rank}\,\mathtt{M} > 3$. If the two videos are unsynchronized, the 4-th singular value is high. [Tresadern+03] uses RANSAC to robustly obtain the synchronization parameters such that the rank constraint is best satisfied, i.e. to obtain a small as possible 4-th singular value. [Rao+03] introduces a method based on epipolar geometry in order to establish temporal correspondence between the frames of two videos. Because of the noise sensitivity of fundamental matrix, the authors propose the use of rank constraints of corresponding points in two views to measure the similarity between trajectories. Given a sufficient number of point matches, an unknown fundamental matrix $\mathtt{F}$ can be computed using the following equation ([Hartley+04, chapter 10])

$$\mathtt{A}\mathbf{f} = 0 \tag{3.8}$$

where $\mathbf{f} = (f_{11}, f_{12}, f_{13}, f_{21}, \cdots, f_{33})^\top$ (i.e. the 9-vector made up of the entries of $\mathtt{F}$ in row-major order) and $\mathtt{A}$ is observation matrix constructed using coordinates of points of two 2D trajectories. Since Eq.(3.8) is a homogeneous equation, for a solution of $\mathbf{f}$ to exist, matrix $\mathtt{A}$ must have rank at most eight. Due to the noise and inexact correspondence, the rank of $\mathtt{A}$ may not be exactly eight. The 9-th eigenvalue increases dramatically when two trajectories are not synchronized. Therefore, the smallest 9-th singular value of $\mathtt{A}$ corresponds to the best match of trajectories. The authors use this rank constraint as the alignment error to synchronize two videos.

The above approaches requires that the fields of view of cameras intersect. [Caspi+02a] copes with the more complex case in which the fields of view do not necessarily overlap, but still requires that the cameras are jointly moving. Furthermore, the cameras have the same center of projection but different 3D orientation and the calibration parameters are unknown but fixed along the sequences. This approach is based on "frame-to-frame" transformation within each sequence. Generally, frame-to-frame transformation $\mathtt{P}$ can be either homography or affine transformation or fundamental matrices. The distance

between two frame-to-frame transformation $d(\mathtt{P}, \mathtt{P}')$ is described as the following formula

$$d(\mathtt{P}, \mathtt{P}') = \frac{(\text{eig}(\mathtt{P}))^\top \text{eig}(\mathtt{P}')}{||\text{eig}(\mathtt{P})||||\text{eig}(\mathtt{P}')||} \tag{3.9}$$

where $\text{eig}(\mathtt{P})$ and $\text{eig}(\mathtt{P}')$ are vectors of eigenvalues of $\mathtt{P}$ and $\mathtt{P}'$, respectively. In other words, Eq.(3.9) is cosine of angle between 2 vectors, and $d(\mathtt{P}, \mathtt{P}') \leq 1$. The method assumes that $\alpha = 1$. Let Δt be the offset time between 2 sequences:

$$d(\mathtt{P}_t, \mathtt{P}'_{t+\Delta t}) = 1. \tag{3.10}$$

Employing this property, the temporal synchronization Δt is recovered by maximizing the following objective function

$$\max_{\Delta t} \sum_t d(\mathtt{P}_t, \mathtt{P}'_{t+\Delta t}). \tag{3.11}$$

Besides, [Spencer+04] uses the frame-to-frame motion of the sequences instead of pixel based comparison between the two sequences in order to recover the temporal alignment. Assume that the two cameras are rigidly joined together and there is non-uniform motion (no constant motion) of the two cameras. A larger motion in one sequence should correspond to a larger motion in other sequence. The maximum correlation coefficient of the motion property occurs at the best temporal offset. The authors propose three measures based on translation and roll rotation in order to recover the temporal alignment. In the matrix representation, an affine transformation $\mathtt{B}$ and homography $\mathtt{H}$ are

$$\mathtt{B} = \begin{bmatrix} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } \mathtt{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}. \tag{3.12}$$

The translation magnitude measure is defined as follows

$$tm_{\text{Affine}} = \sqrt{b_1^2 + b_2^2}, \tag{3.13}$$

$$tm_{\mathtt{H}} = \sqrt{\frac{h_{13}^2 + h_{23}^2}{h_{33}^2}}. \tag{3.14}$$

for the affine model and the 3×3 homography, respectively. The second measure is translation direction based on the idea that the relative direction depends on the relative orientation of the two cameras

$$dir_{\text{Affine}} = \tan^{-1}(b_1/b_2), \tag{3.15}$$

$$dir_{\mathtt{H}} = \tan^{-1} \frac{h_{23}/h_{33}}{h_{13}/h_{33}}. \tag{3.16}$$

The third one is roll measure (rotation about the optical axis) when there is significant rotation

$$roll_{\text{Affine}} = a_2 - a_3, \tag{3.17}$$

$$roll_{\mathtt{H}} = h_{12} - h_{21}. \tag{3.18}$$

Finally, a combination of the translation magnitude and roll motion measure works for a general motion. According to experiments, this paper shows that the similarity measure

in [Caspi+02a] cannot handle inaccuracies in the frame-to-frame transformation and the combination measure can get better result than using only one measure.

Moreover, [Gaspar+14] suggests an approach, in which, instead of explicitly using the features to solve the synchronization, the camera poses using SfM algorithm are exploited. Since the cameras are stationary or jointly moving, the relative inter-camera extrinsic parameters are constant, i.e. there exists a constant rotation matrix $\mathtt{R}$ and a constant translation vector $\mathbf{t}$ that transform coordinates expressed in the reference frame of the first camera into the one of the second camera

$$\mathtt{R}_1^t \mathtt{R} = \mathtt{R}\mathtt{R}_2^{t'} \tag{3.19}$$

$$\mathbf{t}_1^t = \lambda \mathtt{R}\mathbf{t}_2^{t'} + (\mathtt{I} - \mathtt{R}_1^t)\mathbf{t} \tag{3.20}$$

where $(\mathtt{R}_i^t, \mathbf{t}_i^t)$ denotes the camera pose of the i-th camera at frame (time) $t \in T$ and λ is non-negative constant scalar factor. Using the quaternion representation of rotation matrix $\mathtt{R}$ by 4-vector $\mathbf{q}$, the synchronization problem is written in the form of a minimization problem

$$\underset{\Delta t, \mathbf{q}, \mathbf{t}, \lambda}{\arg\min} \sum_{t \in T} \left(\mu_{\mathtt{R}} ||\boldsymbol{\epsilon}_{\mathtt{R}}(\Delta t, \mathbf{q})||^2 + \mu_{\mathbf{t}} ||\boldsymbol{\epsilon}_{\mathbf{t}}(\Delta t, \lambda, \mathbf{q}, \mathbf{t})||^2 + \mu_{\mathbf{q}} (\mathbf{q}^\top \mathbf{q} - 1)^2 \right) \tag{3.21}$$

where $\mu_{\mathtt{R}}, \mu_{\mathbf{t}}$ and $\mu_{\mathbf{q}}$ are positive weighting coefficients, $\boldsymbol{\epsilon}_{\mathtt{R}}$ and $\boldsymbol{\epsilon}_{\mathbf{t}}$ are cost functions that are derived from the constraints in Eq.(3.19) and Eq.(3.20), respectively; and the last term in this expression enforces $||\mathbf{q}||$ to be the unit (more details in [Gaspar+14]).

3.2.2 Independently moving cameras

There are also some works that solve the synchronization problem with free camera motion. In other words, they do not impose any condition on the relative position of two cameras but require the intersection in the field-of-view between cameras. [Tuytelaars+04] requires the correspondences between features tracked in two video sequences. Although the scene points may be moving non-rigidly, for perfectly aligned sequences, these points can be considered as a rigid configuration. The authors compute a rigidity measure of 5 non-rigidly corresponding pair of trajectories tracked throughout the two sequences: four scene points define the world coordinate system, fifth scene point provides a cue for synchronization. Minimizer configuration of this measure corresponds to the time delay between the two video sequences. Besides, [Yan+04] introduces an approach based on the distribution of space-time interest point. A histogram over time is built for each video sequence. And a frame offset is achieved by maximizing cross-correlation between two histograms.

[Meyer+08] presents a two-step algorithm that leads to sub-frame accuracy synchronization based on motion trajectory correspondences. First, a frame accurate time offset is estimated by detecting local extrema of trajectories and matching their characteristic time patterns. The local extrema on trajectory of a moving object in the first sequence can be found in the second sequence as well. Such points provide sufficient information for video synchronization. Second, by estimating a fundamental matrix between the two cameras from nine non-rigidly moving point correspondences, sub-frame accuracy offset is obtained.

[Gaspar+14] deals with the most general and complex case when there is no condition on the relative position of the two cameras. The cameras observe different parts of a common moving object and a second object, typically a static background. The idea is to track two sets of feature trajectory in each camera: one on the common moving object and another on the static background. And the motion of the two objects with respect to each camera is estimated by using a SfM method. Using these results, the motion of moving object with respect to the static background that does not depend on the camera motion, is estimated and provides information for synchronization.

3.3 Estimation of extrinsic parameters

We review the related works on extrinsic calibration. Here we assume that the cameras are synchronized and global shutter, and their inter-camera poses are constant.

3.3.1 Using calibration objects

With an overlapping multi-camera system, the calibration patterns [Tsai86] such as chessboard, circular dots or circular landmark, can be used in order to estimate not only intrinsic parameters but also the inter-camera poses with high accuracy. However, the drawback of this approach is to require the calibration board to be entirely within the field of view of the cameras. This approach may be prohibitive in the practical applications with small overlapping (our case) or with non-overlapping field of views (FoV). To overcome this problem, [Ikeda+03] presents a complicated approach for Ladybug using a calibration grid and a laser measurement system. In this work, the calibration board is put in front of each camera separately. In the meantime, a laser measurement system is used to determine the 3D coordinate of the grid's corners. Employing this geometrical information and point correspondences over images of the grid in different cameras, calibration parameters are determined. [Li+13] introduces a descriptor-based board. With the proposed pattern, the calibration only requires neighboring cameras to see parts of the calibration board at the same time. The disadvantage of this calibration technique is to require large calibration patterns for multi-camera systems.

[Kumar+08] presents a method in order to calibrate for a multi-camera system with non-overlapping or barely overlapping views. To overcome the non-overlapping challenge, an addition mirror is used to allow all cameras to see a common calibration object such as chessboard. The internal and external parameters of a set of mirrored camera poses are found out using classical calibration methods. From the constraints between mirrored camera poses, the authors estimate the external parameters of real camera by solving a linear problem that requires at least five mirrored images. Inspired by this work, [Lébraly+10a] extends by using planar mirror with an unknown geometry scene instead of chessboard, and embeds the multi-cameras rig on a vehicle, for visual navigation purpose in urban environment. The authors use the "averaging" transformation in order to estimate the translation and rotation. Finally, the two above methods use bundle adjustment to determinate the extrinsic parameters corresponding to the real poses of the cameras by minimizing the reprojection error.

3.3.2 Self-calibration

3.3.2.1 Observation of a same scene in different cameras

Besides the approaches use the calibration object and require expert supervision, some researches present a general method without requirement for specific calibration patterns and which is based on natural features in the environment [Li+05], [Solà+08]. Under the central assumption, the relative camera poses differing only by a rotation are achieved in unprepared natural or artificial scenes in [Li+05]. This semi-automatic solution receives as input image point correspondences only and a three-step procedure sequentially determines the center of distortion, the individual intrinsic parameter of each camera, and the relative camera orientation. In [Solà+08], a central EKF-SLAM (Extended Kalman Filter - Simultaneous Localization And Mapping) fuses the information coming from a stereo camera rig to obtain the extrinsic parameters.

Under the assumption that the intrinsic camera parameters and the vehicle's speed are known, an on-board two-cameras system without overlapping field of view is calibrated in [Lamprecht+07]. In this work, the object (e.g. traffic sign) is tracked successively in each camera. A motion of the car is pure translation along an axis with constant speed, i.e. the multi-camera poses are known. The problem reduces to estimate the 6 extrinsic parameters and 3D points. The 3D points are tracked while they are inside the FoV of the first camera at different times. When these points enter the FoV of the second camera, they are tracked at new different times. The authors track three points five times per camera and solve the system of nonlinear equations using least-squares approximation. The extrinsic parameters between two cameras are estimated just by straight driving with constant speed. However, this method is yet unable to estimate the third parameter of rotation and the translation with an accurate precision according to authors.

[Carrera+11] proposes a method to estimate the extrinsic calibration for overlapping as well as non-overlapping multi-camera system. Individually for each camera, a monocular SLAM algorithm estimates the camera motion and builds a 3D map of visual feature locations up to scale. A similarity transformation is robustly estimated between two reconstructions using image matching for 3D points in different reconstructions, a 3-point RANSAC algorithm and bundle adjustment optimization. The calibration procedure requires the cameras to observe common parts of a scene as the robot makes a full 360º turn. This requirement is to ensure that the similarity transformation works by enforcing the global consistency of the maps.

Cannelle et al, publishes papers addressing a panoramic-based calibration method [Cannelle+10], [Cannelle+12]. [Cannelle+10] presents a method in order to calibrate a camera from panoramas. In this framework, all images are acquired from the same point of view and with an overlap ratio around 50%, so they only need to estimate a rotation between images. After that, a bundle adjustment procedure is implemented to compute the pose of images in a panorama (rotation) and calibration camera (intrinsic parameter). In [Cannelle+12], they discuss a method to calibrate a multi-camera head with 10 full HD cameras on a mobile mapping system equipped with an Inertial Navigation System (INS). A bundle adjustment is run in order to jointly estimate the position and rotation of camera relatively to the vehicle/INS, the position and rotation of vehicle relatively to the ground and the 3D scene points relatively to the ground.

3.3.2.2 Camera motion based method

Besides the approaches based on the point correspondences established between the different cameras, an indirect approach (which works even in the case where the cameras have barely overlapping or completely non-overlapping views) is investigated by relying on motion. [Nyman+10] presents a method based on motion of multi-camera rig in which two pure translation motions in different directions are sufficient to linearly recover the rotation component, then two general motions including both translation and rotation are sufficient to linearly determine the translation component.

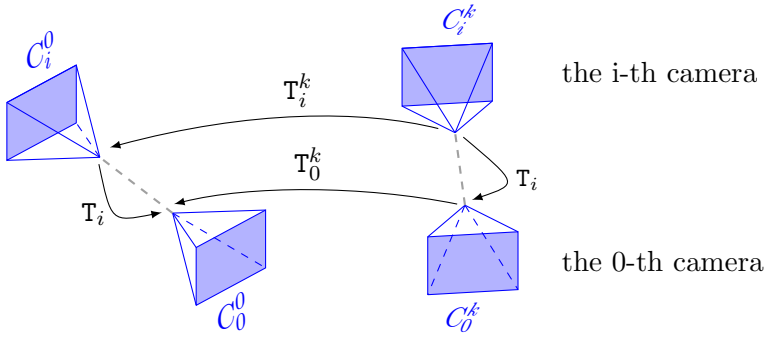


FIGURE 3.2: Relation between cameras in the rig.

The authors in [Luong+93], [Esquivel+07], [Oskiper+07], [Kim+08], [Lébraly+10b], [Heng+13] share the same point start. They compute the displacements of each camera in the coordinate system of itself (for example, by SfM). To cope with the extrinsic calibration problem, the idea of their method is to use the commutativity of the diagram in Figure 3.2 where $\mathtt{T}_0^k(\mathtt{R}_0^k, \mathbf{t}_0^k)$ and $\mathtt{T}_i^k(\mathtt{R}_i^k, \mathbf{t}_i^k)$ are the displacement of the first camera and the i-th camera at the k-th frame, in the coordinate system of the first camera and the i-th camera, respectively; and $\mathtt{T}_i(\mathtt{R}_i, \mathbf{t}_i, \lambda_i)$ is the transformation from the i-th camera to the first camera with λ_i accounts for the different scales of coordinate systems. If the relative displacement from a camera to another camera does not change, this rigidity assumption and simple changes of basis lead to:

$$\mathtt{T}_0^k \mathtt{T}_i = \mathtt{T}_i \mathtt{T}_i^k. \tag{3.22}$$

Eq.(3.22) can be decomposed into the two following constraints: the first one regarding only orientations and the second one linking both orientations and positions

$$\mathtt{R}_0^k \mathtt{R}_i = \mathtt{R}_i \mathtt{R}_i^k, \tag{3.23}$$

$$\mathtt{R}_0^k \mathbf{t}_i + \mathbf{t}_0^k = \lambda_i \mathtt{R}_i \mathbf{t}_i^k + \mathbf{t}_i. \tag{3.24}$$

The problem is that we obtain a different relative pose at each instant. Then we should determine the optimal relative poses. For the translation $\mathbf{t}_i$, it is easy to calculate the mean value (once the rotation $\mathtt{R}_i$ is known), but it is more complicated for the rotation. The orientation constraint in Eq.(3.23) is the same as the hand-eye coordination problem [Park+94] in which a closed-form least square solution is derived. The rotations $\mathtt{R}_0^k$ and $\mathtt{R}_i^k$ are measured in different coordinate frames. The problem of rotation averaging is to find a rotation $\mathtt{R}_i$ in a way that is most consistent with all rotation pairs $(\mathtt{R}_0^k, \mathtt{R}_i^k)$.

Rotation averaging strategies are applied in [Kim+08] in which the authors make a non-overlapping spherical camera assumption (and the cameras share a single optical center) in order to calibrate the multi-camera system using motion estimation. A method for calibrating the rotation between two-camera rig without overlapping FoV using rotation averaging strategies are discussed in [Dai+09]. The issues such as convergence and optimality of rotation averaging algorithms are investigated in order to achieve a global solution for calibration problem. More recently, [Hartley+13] summarizes the research that has been carried out rotation averaging; as well as provides proofs of convergence in many cases.

Let $\mathbf{q}_0^k, \mathbf{q}_i^k, \mathbf{q}_i$ be the unit quaternion representations corresponding to $\mathtt{R}_0^k, \mathtt{R}_i^k, \mathtt{R}_i$, respectively. Eq.(3.23) may be written as following linear equation

$$\mathtt{A}_k \mathbf{q}_i = \mathbf{0} \tag{3.25}$$

where $\mathtt{A}_k$ is 4×4 matrix formed by the elements of $\mathbf{q}_0^k, \mathbf{q}_i^k$. Given the value of $\mathtt{R}_i$, Eq.(3.24) simplifies to

$$\mathtt{B}_k \mathbf{t}_i - \mathbf{c}_k = \mathbf{0} \tag{3.26}$$

where $\mathtt{B}_k = \mathtt{R}_0^k - \mathtt{I}$ and $\mathbf{c}_k = \lambda_i \mathtt{R}_i \mathbf{t}_i^k - \mathbf{t}_0^k$. In 2007, [Oskiper+07] and [Esquivel+07] propose independently the same idea. Their approach first estimates initial parameters $(\mathtt{R}_i, \mathbf{t}_i)$ then refines these parameters by minimizing a suitable cost function using LM framework. [Oskiper+07] works on a multi-stereo system with two pairs of backward and forward looking stereo camera with inertial measurement unit (IMU). The authors suppose that each stereo camera is calibrated (intrinsic and extrinsic parameters) and $\lambda_i = 1$. The goal is to determine the relation pose between two stereo cameras. The set of stereo camera poses is estimated independently using visual odometry (SLAM) in a feature rich environment. RANSAC process randomly selects 3 pairs of corresponding poses for each stereo camera. First, they estimate the unit quaternion that minimizes $\sum_{j=1}^{3} ||\mathtt{A}_{k_j} \mathbf{q}_i||^2$. The solution of $\mathbf{q}_i$ in each RANSAC trial is obtained as the unit eigenvectors corresponding to the smallest eigenvalue of the matrix $\mathtt{A}^\top \mathtt{A}$ where $\mathtt{A} = [\mathtt{A}_{k_1}^\top \mathtt{A}_{k_2}^\top \mathtt{A}_{k_3}^\top]^\top$. Second, from the obtained solution of $\mathtt{R}_i$, the relative translation $\mathbf{t}_i$ is estimated by minimizing $\sum_{j=1}^{3} ||\mathtt{B}_{k_j} \mathbf{t}_i - \mathbf{c}_{k_j}||^2$. In other words, they solve for the $\mathbf{t}_i$ which minimizes $||\mathtt{B}\mathbf{t}_i - \mathbf{c}||^2$ where $\mathtt{B} = [\mathtt{B}_{k_1}^\top \mathtt{B}_{k_2}^\top \mathtt{B}_{k_3}^\top]^\top$ and $\mathbf{c} = [\mathbf{c}_{k_1}^\top \mathbf{c}_{k_2}^\top \mathbf{c}_{k_3}^\top]^\top$. Finally, the best hypothesis determined by RANSAC method is then refined by LM method by minimizing the objective function that is derived from the constraints in Eq.(3.23) and Eq.(3.24) with $\mathtt{R}_i = \mathtt{R}_i(\mathbf{q}_i)$ (multiplying the left-hand-side of Eq.(3.23) by the one of Eq.(3.24) and the right-hand-side of Eq.(3.23) by the one of Eq.(3.24)):

$$\min_{\mathbf{q}_i, \mathbf{t}_i} \sum_k \left|\left| (\mathtt{R}_i \mathtt{R}_i^k)^\top (\mathtt{R}_i \mathbf{t}_i^k + \mathbf{t}_i) - (\mathtt{R}_0^k \mathtt{R}_i)^\top (\mathtt{R}_0^k \mathbf{t}_i + \mathbf{t}_0^k) \right|\right|^2. \tag{3.27}$$

The work in [Esquivel+07] considers a multi-camera system (without IMU). A monocular SfM is performed to each camera in order to determine a trajectory for each. The unknown parameter $\mathtt{R}_i$ is obtained by solving directly the linear equation Eq.(3.25) with one corresponding pose pair of each camera (without RANSAC) if the camera motion is not pure translation. Given $\mathtt{R}_i$, $(\mathbf{t}_i, \lambda_i)$ are estimated from Eq.(3.26) with at least two corresponding pose pairs of each camera. And then a nonlinear refinement is used to estimate simultaneously the relative rotation, translation and scale by minimizing the error function

$$\min_{\mathbf{q}_i, \mathbf{t}_i, \lambda_i} \sum_k (||\mathtt{A}_k \mathbf{q}_i||^2 + ||\mathtt{B}_k \mathbf{t}_i - \lambda_i \mathtt{R}_i(\mathbf{q}_i) \mathbf{t}_i^k + \mathbf{t}_0^k||^2). \tag{3.28}$$

This method assumes that the camera poses in initialization step are globally consistent. Thus, the cost function in Eq.(3.28) only optimizes the extrinsic parameters. In practice, the initial camera poses $(\mathtt{R}_i^k, \mathbf{t}_i^k)$ can be inaccurate and should be refined with the extrinsic parameters $(\mathtt{R}_i, \mathbf{t}_i)$ in order to improve the accuracy.

Inspired by [Esquivel+07], [Lébraly+10b], [Lébraly+11] improve the calibration algorithm by extending classical bundle adjustment. The authors consider a multi-camera system in which the cameras are calibrated (known intrinsic parameters), synchronized and rigidly moving. An initial estimate of extrinsic parameters is obtained linearly by solving the hand-eye calibration problem in Eq.(3.23) and Eq.(3.24) whose solutions represent the rotation by a 3×3 orthogonal matrix instead of unit quaternions. [Lébraly+10b] presents complete solutions for general and singular motions. The SfM algorithm in the case of multiple cameras with constant extrinsic parameters is applied in order to reconstruct multi-camera poses and 3D scene points in the world coordinate system. Finally, the 3D scene points, the trajectory of the multi-camera rig and the extrinsic parameters are refined by using the analytical *multi-camera bundle adjustment* algorithm detailed in [Lébraly+11]. However, the reprojection error is in the undistorted space of the classical polynomial distortion model [Sturm+11]. This is due to the fact that the forward-projection of this camera model does not have a closed-form. We will discuss this method in more details in Chapter 5.

The BA for omnidirectional multi-camera in [Schneider+13] deals with points at infinity, and uses ray directions as observation. The uncertainty which enables a Maximum-Likelihood estimation, is transfered from the measure space to the ray space. The calibration procedure requires the multi-camera system with non-overlapping FoV to be rotated within the scene such that the corresponding points are visible in different cameras at different times. The authors show in experiments that the scene points at infinity can stabilize the estimation of the camera rotations significantly. The refinement of intrinsic parameters is left as future work in [Lébraly+11], [Schneider+13].

[Habib+11], [Wang+12] introduce a procedure for the calibration of a mobile mapping system that integrates multiple sensors for acquisition of images, locations, orientation such as GPS, IMU, etc. First of all, the camera poses (rotation $\mathtt{R}_i^k$ and translation $\mathbf{t}_i^k$) of the k-th frame for the i-th camera in a same global coordinate system are determined through a bundle adjustment. The second step, in each time instant, the geometric relationship of the cameras is calculated as following, for example between pair of cameras $(0, i)$:

$$\mathtt{R}_i(k) = (\mathtt{R}_0^k)^\top \mathtt{R}_i^k \tag{3.29}$$

$$\mathbf{t}_i(k) = (\mathtt{R}_0^k)^\top (\mathbf{t}_i^k - \mathbf{t}_0^k). \tag{3.30}$$

An averaging process is performed in order to obtain mean values for relative parameters. [Habib+11] also presents another procedure by exploiting the invariant geometric relationship $(\mathtt{R}_i, \mathbf{t}_i)$ among the i-th camera and a reference camera (e.g. the first camera, without loss of generality). Let $\mathbf{X}_w$, $\mathbf{X}_i$ be vectors of a same 3D scene point in the global coordinate system and in the coordinate system of i-th camera and λ_i be scale factor. At the k-th frame,

$$\mathbf{X}_w = \mathbf{t}_i^k + \lambda_i \mathtt{R}_i^k \mathbf{X}_i, \;\; \mathbf{t}_i^k = \mathbf{t}_0^k + \mathtt{R}_0^k \mathbf{t}_i, \;\; \mathtt{R}_i^k = \mathtt{R}_0^k \mathtt{R}_i. \tag{3.31}$$

Thus

$$\mathbf{X}_i = 1/\lambda_i (\mathtt{R}_i)^\top (\mathtt{R}_0^k)^\top (\mathbf{X}_w - \mathbf{t}_0^k - \mathtt{R}_0^k \mathbf{t}_i) \tag{3.32}$$

and image point $\mathbf{p}_i$ can be expressed as function of $\mathbf{X}_i$ and given intrinsic parameters:

$$\mathbf{p}_i = p_i(\mathbf{X}_i). \tag{3.33}$$

Note that the scale factor λ_i is eliminated through projection (division) process. We can see that the left side of Eq.(3.33) is measured observation, meanwhile, the right side includes all parameters $(\mathbf{X}_w, \mathtt{R}_0^k, \mathbf{t}_0^k, \mathtt{R}_i, \mathbf{t}_i)$. A general *least squares adjustment* procedure is implemented in order to estimate all parameters based on the principle of least squares of observation residuals. The quality of calibration highly depends on the distribution of the control points in the calibration field, hence, the procedure is performed in a special room where hundreds of calibration targets are well distributed along the walls, ceiling and floor.

[Heng+13], [Heng+14], [Heng+15] propose two methods in order to determine the relative geometric relationship of the cameras without assumption of overlapping FoV. In their work, a multi-camera system (a set of four CCD fish-eye cameras integrated into a car body or mounted on a roof rack) provides a surround view of environment. First at all, the cameras are calibrated (intrinsic parameters) using calibration pattern. In [Heng+13] (with improvement in [Heng+15]) introduces SLAM-based self-calibration. Odometry data (commonly available on vehicles) is used to obtain a set of camera motions together which is required as an initial estimate step of the extrinsic. The initial estimate of transformation between camera and odometry is found from camera poses computed by monocular visual odometry (VO) and odometry data. The 3D scene points are reconstructed from inlier feature tracks in monocular VO, odometry data and initial camera-odometry estimate. The camera-odometry transformations and 3D scene points are refined by minimizing the image reprojection error. 3D scene information with camera-odometry transformations and images from the multi-camera system are used in loop closure detection. The vehicle poses are corrected using robust pose graph optimization. The accuracy of inter-camera transformation is guaranteed by finding feature point correspondences across different cameras. In other words, the authors do exhaustive feature matching between each camera's current frame and these in the other camera's frame history. A rectification step is required to ensure a high number of inlier feature point correspondences. This step is computationally expensive. Finally, a BA optimizes all parameters: intrinsic camera parameters, camera-odometry relation, inter-camera relation (extrinsic parameters), vehicle poses (in the world coordinate) and 3D scene points. The cost function involves the sum of image reprojection error from all scene observations and the sum of relative pose error in odometry data measurement. SLAM-based self-calibration has a high computational cost due to finding local inter-camera feature point correspondences and bundle adjustment.

In order to avoid an exhaustive search of inter-camera feature correspondences and loop closures, [Heng+14] (expanded in [Heng+15]) proposes an approach by relying on a pre-existing map. Based on natural features in the environment, a sparse feature map (as a virtual 3D chessboard) is built by SLAM-based self-calibration [Heng+13]. This method does not require a calibrated odometry and allows us to directly estimate the extrinsic parameter with metric scale vie image-based localization. From images of the multi-camera system and the 3D map, the camera poses are estimated. Then, an average estimate of extrinsic parameters is obtained from hand eye calibration Eq.(3.23) and Eq.(3.24) using the quaternion averaging method. In order to optimize the camera extrinsic parameters, the authors solve the non-linear refinement problem minimizing the sum of all reprojection errors. This method depends on a map of the calibration area

created by the complex and expensive method in [Heng+13]. According to the authors, in some occasions, a wrong loop closure, inaccurate odometry data or too few features in the environment led to an inaccurate map, and in turn, inaccurate estimate of the extrinsic parameters.

3.4 Rolling shutter (RS)

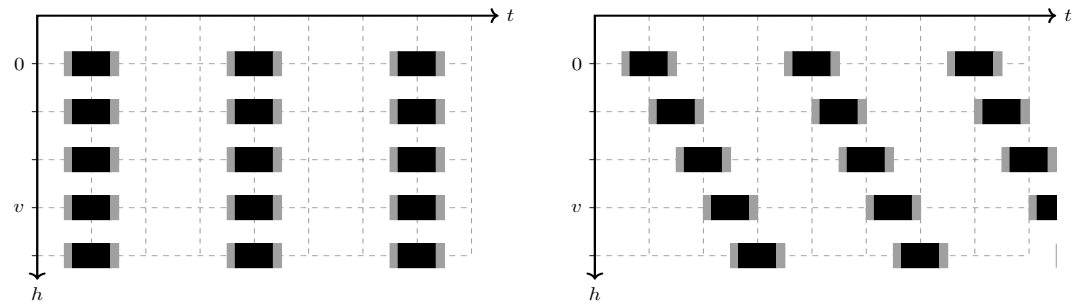
3.4.1 Introduction

The function of a camera shutter is to allow light to pass through for a determined period of time. The used shutter can either be mechanical or electronic, and have a global, block or rolling exposure method. In a global shutter (GS), all pixels in a frame are acquired at a same instance. On other hand, rolling shutter (RS) is a technique used when acquiring images by scanning the frame. Instead of imaging the scene at a single time instance, the scanlines are sequentially captured. Figure 3.3 demonstrates effects of mechanical RS.

FIGURE 3.3: A two-wheeled bobsleigh taking a turn at 60 km/h shot by Jacques Henri Lartigue. This picture is taken with an ICA (Internationale Camera Actiengesellschaft) camera in 1912. The bystander at the side of the road is leaning to the left, while the hind wheel of the car is deformed to an oblique ellipse.

The two most common image sensors used in digital cameras are the CCD (Charge-Coupled Device) and the CMOS (Complementary Metal Oxide Semiconductor) image sensors. Generally, CCD sensors use GS and CMOS sensors use RS. In CMOS sensors with RS, each row of pixels are reset, exposed and read out in sequential order over time (see Figure 3.4). The rows which are not being read out, continue to be exposed.

Due to cost and energy efficiency, the CMOS sensors are gradually replacing CCDs sensors. Almost all camera-equipped cell-phones make use of a RS. In particular, they are able to acquire higher frequency scene dynamics via their intrinsic time-varying intra-image measurements. The RS cameras however generate geometric distortions in the images if the camera is moving or objects move in scene, and the amount of these deformations depend on how fast the shutter rolls and how fast the camera motion is. When a camera equipped with a RS moves quickly relative to the subject of the video, several of artifacts can be found in the acquired video.



(a) Global shutter (b) Rolling shutter

FIGURE 3.4: Global Shutter vs Rolling Shutter. Each block symbolizes row of image.

3.4.2 Distortion effects

There are four major types of RS deformations: skew, wobble, smear and partial exposure. Either due to fast motion or long exposure, another artifact that can be observed in GS camera as well as RS camera is motion blur.

Skewing occurs when the camera undergoes a constant (or smoothly varying) motion. It is worth noting that skew will procedure a shear transformation. An example is shown in Figure 3.5 where the pedestrians appear slanted in image as they move to the left across the scene.

FIGURE 3.5: Walk around Madrid, shot by Tomás G. Santis using Panasonic DMC-G6. Source: www.flickr.com/photos/tgsantis/. The RS goes from top to bottom, the image motion goes from right to left.

Wobble occurs when there are high accelerations or the motion is at a higher frequency than the frame rate of the camera. This artifact is particularly pronounced for cameras mounted on cars or motorbikes when cameras are vibrating. Straight lines in a scene will occur curved and different parts of a single frame can be either compressed and stretched at the same time, for example, Figure 3.6. Wobble deformations incline to be fairly small and as such are often more difficult to notice in still image than in video sequence.

Smear is a higher frequency artifact than wobble, and is essentially an aliasing effect occurring between rows. When this occurs, portions of objects can appear to be floating and disconnected from the rest of the object itself. We usually see smear

FIGURE 3.6: London Eye using a mobile phone camera. Source: www.flickr.com/photos/drinksmachine/.

when an object in the scene is moving at hight speed, rather than the camera itself, for example, rotation propeller in Figure 3.7.

FIGURE 3.7: An airplane propeller shot by Jason Mullins using Apple iPhone. Source: www.flickr.com/photos/jasonmullins/.

Partial exposure occurs when the scene being recorded changes drastically during the frame image. This will result in a two or more distinct portions of the image that look strange when combined. Similar problems can arise with fluorescent lighting, strobe effects, lighting or any extreme situation where very fast motion or very fast bursts of light are seen while the CMOS chip is sequentially recording a frame. Figure 3.8 demonstrates this phenomenon.

Motion blur occurs in GS camera as well as RS camera. Because of technological constrains or artistic requirements, the image (all image-GS or each line-RS) represents the scene over a period of time. Most often we can assume that this exposure time is instantaneous. But this is not always so, and a fast moving or a longer exposure time may result in blurring artifact which make this apparent (see Figure 3.9). Motion blur depends directly on each pixel's exposure period (electronic shutter interval) and even with small motions some amount of blur is present. Subsequently if there is enough motion to produce RS distortions, there is enough motion to create motion blur.

FIGURE 3.8: Lightning conditions changed between the top and bottom parts of the photo. Source: https://en.wikipedia.org/wiki/Rolling_shutter.

FIGURE 3.9: Motion blur, RS and GS. Source: www.teledynedalsa.com.

3.4.3 Rolling shutter parameter

Early work that specially models RS distortions is presented in [Wilburn+04]. The authors use an array of CMOS cameras to create undistorted images by selecting the scan-lines from different cameras but which are acquired at the single instant of time. A first study on SfM from a RS video sequence is introduced in [Meingast+05]. This section will discus this model of RS camera.

In a RS camera, the scanlines are sequentially exposed, read in and immediately sent. The frame rate f (frame/s), the exposure length of one scanline e (μs), the rate at which scanlines are exposed r (scanline/μs) and any delay between frames d (μs) are the variables which control exposure of the scanlines (see Figure 3.10). For IIDC/DCAM cameras (IEEE 1394 interface), the delay d would normally be 0 with a RS; for general cameras, though, this has to be verified. The effect of non-zero e is motion blur within the scanline, but has no geometric effects.

When a RS camera creates an image, each scanline of pixels is scanned in sequential order over time. A camera model is formalized by noting that each scanline corresponds to a different instant in time. [Meingast+05] assumes that the exposure is instantaneous. The important parameters for RS modeling is the time between the start of integration of two consecutive scanlines $\tau = 1/r$, so called *line delay*. The sign of τ depends on whether scanning direction is top-to-bottom or bottom-to-top. Let t_0 be the starting instant of the first exposed scanline. The scanline v will be exposed at instant

$$t(v) = v\tau + t_0. \tag{3.34}$$

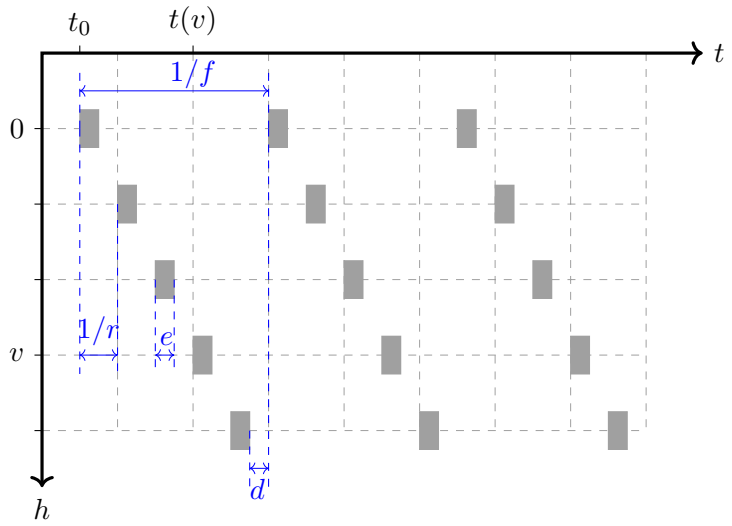


FIGURE 3.10: Rolling shutter model. Each gray block symbolizes the exposure length of one scanline.

To account for those RS effects, the frame time t_0 and line delay τ need to be known. On most RS cameras, each image is captured with a fixed line delay [Meingast+05]. In the next Subsection 3.4.4, several methods for dealing with calibration for RS camera are introduced. Subsection 3.4.5 describes a post-capture processing in order to remove RS artifacts. The previous works on 3D reconstruction for RS camera are mentioned in Subsection 3.4.6 and Subsection 3.4.7.

3.4.4 RS calibration

We discuss the important parameter which allows us to model RS deformations. In this subsection, we review methods that determine the RS line delay.

[Meingast+05] proposes a method to calibrate the RS timing using additional hardware and studies the different RS effects under special fronto-parallel motion. The RS camera is exposed to a LED flashing at high frequency. The resulting image includes light and dark lines whose spatial frequency is linked to the line delay and the known LED frequency. The precise knowledge of the LED frequency is essential for a successful calibration and the authors suggest to remove the lens for best sensor illumination. This method, however, is prone to imprecision, especially if the camera has a fixed lens. [Ringaby+11] refines the method of [Meingast+05] to cope with partly illuminated sensors occurring if the lens is not removed.

Conceptually, the simplest way to calibrate a RS camera in the lab is to first capture an image of a scene with the camera static and then capture a short video of the same scene with the camera undergoing a constant motion [Thalin10]. The RS parameter τ can then be estimated from the motion in the video and an estimate of the skew between the first still image and a frame in the video. The key element in this set-up is that the data contains imagery with two different motions (zero and constant).

In some cases, it is possible to calibrate RS camera in the lab. In many cases, however, all we have is a video obtained from an unknown source. The authors in

[Baker+10] propose an auto-calibration algorithm to estimate τ from a short segment of jittery video. This method requires the video to contain temporally varying motion. Generally speaking, the more jitter, the better the τ estimate. First, they introduce a method aim to remove wobble artifact. Given line delay τ, they analyze how sensitive is their RS rectification to an incorrect setting of line delay τ. Then they conclude that, in theory, with an incorrect value of τ, the applied correction only differs from the one that would be obtained with the correct value of τ by a slightly different affine warp for each frame. They attempt to detect and minimize the residual affine jitter. RS rectification is performed for a sampling of different values of τ and compute optical flow across each output video, then compute a measure of how "translational" the optical flow is on average across the sequence. The measure (residual affine from frame to frame) across τ is plotted. They smooth the result slightly and then perform the calibration by choosing τ to take a minimum value of the measure.

Besides the line delay calibration proposed by [Meingast+05], [Karpenko+11] proposes a method to estimate the camera's line delay that relies on a gyroscope. In this work, a cell-phone camera and a gyroscope are attached rigidly. The camera with a FoV of 45^o is modeled by pinhole camera model. The camera motion is only described in terms of its rotation $\mathtt{R}(t)$ at time t and is interpolated using Special Linear intERPolation (SLERP) of quaternions [Dam+98] from high-frequency gyroscope data. The unknown parameters including the focal length, the RS parameters, the gyroscope drift and the delay between the gyroscope and the image frame sample are recovered by minimizing the reprojection error of the set of corresponding points between two successive frames.

[Oth+13] proposes an approach to calibrate the line delay using a video sequence of a calibration pattern with known geometry. A continuous-time trajectory model is combined with a RS model to estimate the line delay. First, the intrinsic and distortion coefficients of a RS camera are estimated with GS methods using still images and a calibration pattern. Next, this camera is moving in front of the known pattern and produces RS deformation. The authors model the trajectory of the camera in continuous time domain - *continuous-time batch optimization technique*. The pose of the camera is parametrized as a fourth-order B-spline. A smoothing spline is initialized using GS perspective-n-point (PnP) solution at each image time. They show that the number of B-spline knots affects the stability of line delay estimate and RMS errors. So an adaptive knot placement, that needs to represent different parts of the trajectory to avoid over-fitting, is proposed. They also introduce a linear approximation of the covariance matrix of reprojection error terms for the RS camera model. The non-linear least-squares problem is solved using DogLeg method. This method is experimentally evaluated more accurate than the technique for calibrating the line delay (which requires specialized hardware) in [Meingast+05].

3.4.5 Video rectification and stabilization

In this subsection, we address to post-capture image rectification and stabilization of a video. There exists a number of different approaches to remove RS artifacts in special cases.

[Liang+08] deals with skewing effect. The authors focus on the scenario in which the camera undergoes with respect to the object a translational motion parallel (or near parallel) to the image plane. Bézier curve (3 degree polynomial) is chosen to produce

smooth representation of camera or object trajectory. The global motion (translation) is found as the peak in a 2D histogram of translation vectors obtained using block matching. This method cannot handle the scenarios with large depth range or multiple objects having different motions. [Baker+10] copes with wobble artifact under large accelerations or jitter. The authors extend the method in [Liang+08] based on translational motion model by replacing Bézier interpolation with L1 regularization across scanlines, allowing for more general motions: affine motion and low-frequency motion.

Partial exposure artifact is investigated in [Bradley+09]. The authors present two simple methods to solve both RS shear and synchronization at the same time. The first approach is based on strobe illumination that results in images: darker in some regions and lighter in others. By locating the exposure flash in each camera, the synchronized frames are identified. Its applicability is limited to indoor environments. The second approach, while being less accurate than the previous, is applicable to more general illumination conditions by employing a sub-frame wrap along optical flow vectors.

[Forssen+10], [Ringaby+11] assume that the RS parameter, line delay τ is known using the method proposed in [Meingast+05] with a modification. Once camera motion is known during a RS exposure, it is used to rectify the frame. The authors employ spherical linear interpolation (SLERP) to estimate intermediate rotations. Applying them to the image point, all rows are rectified to the first row. This rectification algorithm also allows for video stabilization by smoothing the estimated camera trajectory.

[Grundmann+12] presents an algorithm for removal of RS distortions in uncalibrated streaming video. The authors partition the image domain in m blocks, resulting in m unknown homographies $\mathtt{H}_k$ $(k = 1, \ldots, m)$ which need to be estimated per frame to model the RS distortion by using the normalized direct linear transformation [Hartley+04]. They express the RS deformation parametrically as homography mixtures: for point $\mathbf{x}$, $\mathtt{H}_\mathbf{x} \doteq \sum_{k=1}^{m} w_k(\mathbf{x})\mathtt{H}_k$ where $w_k(\mathbf{x})$ is a Gaussian weight centered around the middle of each scanline block k. Their homography mixtures are regarded as interpolation techniques to local homographies with additional regularization for improved stability. To evaluate their results qualitatively and compare to the results of others methods, they use a user study based on subjective opinion of participants.

3.4.6 Perspective-n-point problem

The Perspective-n-Point (PnP) problem refers to the absolute pose estimation problem and has a great importance in performing robotics visual SLAM, localization with respect to a given map, and SfM. While there is an inherent difficulty in the absolute pose estimation with RS cameras, there is already a widespread usage of RS cameras. This problem has received much attention but is not the thesis topic. In this section, we briefly address to previous works on PnP for RS camera(s).

[AitAider+06] for the first time stops considering the RS artifacts as drawback and exploits them to simultaneously extract the pose (6 DoF) and velocity (6 DoF) of an object relative to the camera frame from a single view. Using a known 3D model, the authors propose the first PnP solution for RS camera under linear motion through iterative minimization. Moreover, [AitAider+09] shows how one pair of RS images enables the computation of both structure and velocity of rigid moving objects. The authors first compute an initial estimate of the 3D points neglecting RS effect and then refine

the shape and velocity using a nonlinear least squares algorithm (LM). As an alternative to iterative approaches, [Magerand+12] proposes an approach for global optimization of pose and dynamics from a single RS image using Taylor expansion of Rodrigues formula for rotation and elimination of the translational parameters. However, this approach is heavily sensitive to mismatches and it is not a feasible solution for a RANSAC loop due to long run time.

Recent work in [Albl+15] presents a two-step algorithm. The authors first apply a standard P3P algorithm for GS model to estimate poses and then refine by their R6P solver based on RS model and a small rotation approximation. [Albl+16a] extends the solution R6P to R5Pup, "-up" denoted up-vector information. Instead of using the initialization by a standard P3P, the camera absolute pose directly can be estimated from five 2D-to-3D matches using IMU information (the rotations around 2 axes are known). The problem is to estimate 10 parameters: 3 for translation, 1 for rotation and 6 for velocity. Five 2D-3D correspondences are necessary to solve it. Another minimal solution to the RS pose and translation estimation problem is introduced in [Saurer+15] using five 2D-3D correspondences. Under the assumption of a constant linear translational velocity (the rotational velocity of camera is zero), the solution for all the 9 unknown parameters (3 DOF for rotation, for translation and for translational velocity) is based on Gröbner Basic. Finally, the authors relax the assumption and apply LM method to refine full velocity and pose of the RS camera.

[Dai+16] shows that the usual 3×3 essential matrix for pinhole model does not exist for RS camera. The authors derive a generalized epipolar geometry: a 5×5 generalized essential matrix for a pure translation RS camera with a constant translation velocity and a 7×7 generalized essential matrix for a RS camera with a constant angular velocity and a constant linear translation velocity. This algorithm can be integrated into RANSAC approach. Given the generalized essential matrix, the relative transformation between two images (i.e. between the poses of their first scanlines), the translation velocity and/or rotation velocity can be recovered using matrix factorization.

3.4.7 Structure-from-Motion

In this subsection, we address to previous works on SfM problem for RS camera(s). [Hedborg+11] proposes a RS-specific SfM algorithm. The camera trajectory is described by linearly interpolating between the camera poses at the beginning of each frame using SLERP. Considering the observations, the most significant RS deformations occur for rotational motion, according to the authors. They suggest to invert the projection equations with a rotation-only assumption to undistort the frames and then apply a GS SfM algorithm. The preliminary rectification of the frames significantly improves the results of reconstruction. In several cases, they show that the accuracy of reconstruction depends on the rectification step and any model errors in this step will also propagate to the final reconstruction. Furthermore, this rectification step ignores the important translational velocity information contained in the RS camera. Their approach is generalized in [Hedborg+12]. They remove the rectification step to propose the first RS BA. The pose between 2 consecutive frames is approximated with a linear interpolation for the position and SLERP for the rotation parameters. They adapt the BA equations and propose the triangulation and PnP steps for RS cameras. For solving the perspective

pose problem, they suggest to use a multi-frame PnP solver and simultaneously estimate the pose of the camera in multiple frames. Experimentally, the direct use of the RS images improves the performance and stability of the pose estimation.

A SfM pipeline is proposed in [Klingner+13], for cameras mounted on a car, which uses relative pose prior along the vehicle path. The authors consider images instead of video (a repository of billions of 2D images captured with RS cameras rigs including 15 cameras along vehicle trajectories) which make tracking harder, but they have the advantage of starting with a better high-frequency pose prior using GPS and inertial sensors. A generalized camera model based on raxel (ray pixel) or rosette is used for their camera rigs. Intrinsic rosette and RS timings are calibrated. The RS model relates pixel coordinates $\mathbf{p}$ and time as some functions $t(\mathbf{p})$. A fundamental operation in BA is triangulation from multiple views. Given multiple image observation $\tilde{\mathbf{p}}$ of an unknown 3D world point, one wishes to find the world point $\mathbf{X}$ that minimizes reprojection error $||\mathbf{p} - \tilde{\mathbf{p}}||^2$ where $\mathbf{p} = \mathtt{T}(t(\mathbf{p}))\mathbf{X}$ and $\mathtt{T}(t)$ is transformation between world coordinate system and 2D image system at instant t. The constraint is implicit in $\mathbf{p}$, an approximation is proposed with a simple assumption: $t(\mathbf{p}) \simeq t(\tilde{\mathbf{p}})$, because RS tends to be fast and $t(\mathbf{p})$ is slowly changing function. This simplifies the RS problem to that of standard multi-view triangulation. The authors fuse the sensor data (inertial, GPS) to establish an initial trajectory for the vehicle without imagery.

[Meilland+13] performs real-time (incremental) dense structure and motion estimation using a RGB-D sensors (Depth sensor). A unified model attempts to simultaneously correct for both RS and blur motion. The camera is pre-calibrated using the method proposed in [Ringaby+11]. Six parameters for velocity in Lie Group $\mathbb{SE}(3)$ are sufficient for modeling both RS and blur motion. In this work, a direct dense registration (image-based approach) that does not require feature extraction and matching, is more robust than feature-based approach, especially in the case of motion blur. The performance of this approach is evaluated using both ground truth and real data sequences. Their proposed approach improves over previous RS approaches because it handles motion blur.

[Duchamp+15] considers a calibrated camera, i.e. its intrinsic and RS parameters are known. In comparison to the GS model, the authors add 6 parameters (rotational and translational velocities) for each keyframe to avoid the linear interpolation of the pose for each line of the image used in [Hedborg+11]. An initial solution of the camera pose is found using the epipolar constraint. The authors solve a non linear system instead of linear system in GS image. The 3D points are obtained by triangulation. Finally, a BA optimizes not only the camera pose but also rotational and translational speed at every keyframes. The method is experimented on short synthetic and real data.

The work in [Albl+16a] incorporates R5Pup solver (using only five 2D-3D correspondences and 2 known parameters of camera orientation provided by IMU information) in SfM pipeline. The initial geometry estimation is still estimated using GS assumption but immediately optimized using BA with RS model. After that, new added poses are modeled using R5Pup solver. Four parameters of camera pose (2 rotation parameters are known) and six parameters of velocity are optimized throughout BA. However, the reconstruction can fail due to poor precision of the measurements from cellphone IMU. In order to handle this problem, the authors propose an approach to transform already reconstructed scenes into a standard position such that downward direction of new camera pose is as close to vertical direction as possible. Their method is compared to SfM for GS model and is evaluated on real data captured by a cellphone while walking.

The recent study of [Albl+16b] is the first to show *degeneracy* in RS SfM in which the parameters cannot be uniquely determined. The authors use RS model with linearized rotational and constant translational velocity and show that, in many practical situations, images taken by perspective cameras become critical when reconstructed with this model. Without loss of generality, supposing that the y-axis of camera coordinate system is image RS direction (readout direction). A degeneracy occurs if all image poses have parallel y-axis (of camera coordinate system) in world coordinate system. In their experiments, they show that in order to obtain a correct reconstruction using RS SfM algorithm, the input images should be captured with different readout directions, for example, two image sets with perpendicular readout direction. For the same purpose, the work in [Ito+17] explains the same degeneracy but in a different way. They consider RS camera model with pure rotation with constant angular velocity and propose an approximate RS camera model - affine camera approximation, i.e. the first-order approximation of perspective effect on RS camera. The rotation is also parametrized by axis-angle representation. The RS parameters are rotational velocities $[\omega_1, \omega_2, \omega_3]$. They show that the SfM of RS camera is equivalent to self-calibration problem of an image sequence that has unknown, varying skew and aspect ratio along with varying lens distortion of a special kind. Particularly, the aspect ratio is ω_1 (rigorously, $1 - \omega_1$) and ω_2 is unknown and varying skew in the image sequence; and ω_3 can be treated equally to radial distortion. For this model, the general representation of degeneracy reported in [Albl+16b] is derived. When this critical motion occurs, the estimation of ω_1, ω_2, y-component of camera pose for each image and y-component of 3D points are ambiguous by a scale. Besides avoiding this critical motion, they can deal with by determining somehow either ω_1 or ω_2 for a selected single image. A simple way is to find out an image in the sequence that has as small distortion as possible or free-distortion and set these parameters to zero. The ambiguous scale is solved. The proposed approach is evaluated by experimental results on synthetic and real data.

3.5 Self-calibration and synchronization of sensors

In the context of a general multi-sensors, [Furgale+13] simultaneously estimates the temporal and spatial registrations between sensors. In the experiments, the multi-sensor is composed of a camera and IMU. The best accuracy is obtained thanks to the use of all measurements at once, a continuous-time representation (a B-spline for IMU poses) and maximum likelihood estimation of the parameters (time offset, transformation between IMU and camera, IMU poses and others). The authors show that it is better to calibrate jointly time offsets and relative sensor transformation than to determine these parameters separately.

SLAM on RS camera and IMU is done in [Lovegrove+13], [PatronPerez+15] using a sliding window batch estimation of the continuous camera trajectory. In these papers, the authors also investigate the use of their framework to estimate relative pose, bias and the camera focal length. A continuous-time pose representation makes use of cumulative cubic B-splines in the Lie group $\mathbb{SE}(3)$. According to the authors, using a cumulative B-spline not only preserves $\mathcal{C}^2$ continuity but also provides a simple second derivative formulation that is useful for generating inertial predictions. This parameterization, when applied locally, is free from any singularities and offers a good analytical approximation to smooth trajectories. Reprojection errors are computed between the first measurement of a pattern (reference pose) and subsequent observations (measurement pose) and are

minimized using non-linear least squares method (LM framework). As the papers use an expensive IMU with GPS-clock synchronization, no time synchronization is needed. The RS parameter is also pre-calibrated.

[Vo+16] synchronizes and self-calibrates consumer cameras (Gopro, iPhone 6, Samsung Galaxy 6) using BA in a context different to ours: free camera motion (i.e. independently moving cameras), assuming that the resulting sequences have significant overlap. So, this method requires the corresponding 2D trajectories across cameras. The authors use static scene reconstruction of the scene background, dynamic scene reconstruction of the scene foreground by jointly optimizing the spatiotemporal camera parameters (intrinsic parameters, camera poses, time offset between cameras) and static and dynamic 3D scene. The spatiotemporal cost function involves image reprojection cost and physics-based motion priors cost for moving objects. The RS is not considered in this work.

3.6 3D reconstruction

In this section, some previous works on 3D reconstruction from omnidirectional multi-cameras are mentioned. The state-of-the-art of the image based surface reconstruction is reviewed in [Litvinov15]. The authors introduce a sparse, incremental and 2-manifold surface reconstruction from a sparse SfM points cloud based on 3D Delaunay triangulation. More precisely, this method receives as input a sparse 3D point cloud generated by SfM. For every new frame, the surface is updated in a small neighborhood of the new frame. It allows us access to results during the processing with low and constant memory consumption and it is useful to reconstruct large scale scenes. Their method returns a 2-manifold output surface. Such a surface is useful for post-processing and applications (see [Litvinov15] for more details). Their method is experimented on synthetic and real sequences taken by an omnidirectional GS multi-camera.

The authors in [Saurer+13] propose a stereo algorithm that takes into account the RS model and produces geometrically consistent 3D reconstructions. They attempt to make traditional Computer Vision algorithms (e.g. stereo, registration) that work on RS cameras. The camera used in this work is calibrated wide angle RS camera. Significant lens distortions that often present in wide angle camera, intertwine with RS effects when the camera is moving in the scene. The algorithm solves for time of exposure and depth at the same time, even in the presence of lens distortion and produces accurate 3D models from RS cameras. RS needs to be considered only where its effects are significant, i.e. for stereo in the range of one pixel or more. Locally, these deformation errors are not as visible and may seem to be of minor importance. However, for accurate reconstruction from a driving car they are significant. The authors analyze the setting of camera motion inducing RS artifacts and show that even for every moderate speeds and resolutions, these effects are significant.

[Saurer+16] proposes and implements a pipeline for sparse to dense 3D reconstruction from large scale wide baseline images captured by a rig of 15 RS cameras mounted on a moving car. A RS BA which optimizes 3D points, camera poses and velocities, is proposed. The optimization which only models RS effect, can fail due to wide baseline images with poor visual connectivity. To deal with this problem, the cost function involves with GPS/INS information. Furthermore, an additional smoothness term which

enforces pairwise smoothness between neighboring poses, is introduced. A dense 3D structure is computed from the refined poses using a multi-view RS plane sweeping stereo algorithm similar to [Saurer+13] with speed-up improvement. They evaluate their pipeline on a long camera trajectory.

3.7 Conclusion

In this chapter, we have presented the current state-of-the-art of self-calibration for multi-camera system: both intrinsic and extrinsic parameters; synchronization of video sequences; rolling shutter effect and 3D reconstruction from multi-camera systems. A number of approaches are proposed to estimate these system parameters in separate processes in most cases: first estimate time offsets and then solve the spatial transformation between cameras. Some works consider a synchronized multi-camera system and solve only extrinsic parameters. In our work, we demonstrate a frame-accurate synchronization method that deals with cameras with small (or even empty) shared FoV. Our synchronization method compares instantaneous angular velocity estimated by a monocular SfM, which does not have the inconveniences in [Spencer+04]: heuristic (translation) or uncalibrated (homography/fundamental matrix) transformations without radial distortion. Moreover, we improve a multi-camera BA in the literature [Lébraly+11]. Our multi-camera BA refines intrinsic parameters (not only inter-camera poses and the other 3D parameters). We also propose the first self-calibration method designed for (omnidirectional) multi-camera systems including synchronization and rolling shutter. We estimate not only the usual parameters but also subframe-accurate synchronizations and the RS parameter (line delay). Our approach is visual-only and deals with long trajectories (hundreds of meters or kilometers).

Overview of our framework

Overview of our algorithm

In this thesis, we consider 360 degree and spherical cameras built by fixing together several consumer cameras pointing in different directions. We propose a framework that synchronizes and self-calibrates these omnidirectional multi-camera systems.

First, the monocular camera model (we experiment two models) is initialized in Chapter 4, Section 4.1 thanks to assumptions that are suitable to an omnidirectional camera without a privileged direction: the cameras have same setting and are roughly equiangular. We apply monocular SfM and calibration refinement by bundle adjustment (BA).

Second, a frame-accurate (FA) synchronization between all videos is obtained by using a method based on instantaneous angular velocity (IAV) in Chapter 4, Section 4.2. Now, we skip few frames in each video such that the sequels of the videos are FA synchronized: frames with the same index are taken at the same time up to the inverse of frame per second (FpS).

Third, a central multi-camera calibration is initialized from the estimated intrinsic monocular parameters and approximate inter-camera rotations. We apply multi-camera SfM [Mouragnon+09] (see also Section 2.4) followed by multi-camera bundle adjustment (MCBA) [Lébraly+11] by adding the intrinsic parameters as new estimated parameters in Chapter 5. Up to now, we did three approximations: global shutter (GS), central multi-camera and zero subframe residual time offsets. Furthermore, we only applied the SfMs (both monocular and multi-cameras) on the beginning of the videos to obtain initial synchronization and initial calibration (the 2k first frames in our experiments). Then the multi-camera SfM is applied a second time on the whole videos.

Last, we apply the MCBA in Chapter 6 for estimating subframe accurate synchronization (SFA) and line delay between two successive lines in addition to usual parameters. We experiment using videos taken by consumer cameras mounted on a helmet and moving along trajectories of several hundreds of meters or kilometers, and compare our results to ground truth.

Cameras

The consumer multi-cameras are modeled by several rigidly mounted monocular cameras and the user fixes them on a helmet. We assume that all calibration parameters (time offsets, line delay, intrinsic with radial distortion, relative poses - extrinsic parameters) are constant during a video acquisition. The camera gain is not fixed and evolves independently for every camera.

There are 360^o cameras composed of four GoPro Hero 3 camera [GoPro], that are started by a single click on a wifi remote. The user can choose the relative poses of the cameras: they are enclosed in a cardboard for small baseline (see Figure 3.11) or are fixed by using the housings provided with the cameras (larger baseline) in Figure 3.12.

There is also a spherical camera modeled by two opposite fisheyes (no relative pose choice) that are synchronized. The Ricoh Theta S multi-camera [ThetaS] in Figure 3.13 has a very small baseline thanks to the use of a prism mirror in front of a monocular camera (its FoV is split in two equal parts, each of them sees more than a half-sphere as a real fisheyes does).

We also experiment on a professional multi-camera (PointGrey Ladybug 2 [Ladybug2]) in Figure 3.14 since its ground truth is provided by the manufacturer (as a table of rays) and also for experimenting on an ideal multi-camera with GS and perfect synchronization. This camera consists of 5 monocular cameras (the camera pointing towards the sky is unused).

Videos

There are three real multi-camera videos taken under various conditions using four GoPro camera: BikeCity1 (BC1) - bike riding in a city, WalkTown (WT) - walking in a town, FlyHill (FH) - paragliding flying at very low height above a hill. WT is taken in the early morning during summer to avoid moving cars and pedestrians, but lighting is low. BC1 has sunny lighting (with contre-jours - French for against daylight) and most cars are parked. WT and BC1 have the same calibration setting. FH has larger baseline and lower FpS and better angular resolution. Figure 3.11 shows images and cameras of BC1 and WT, Figure 3.12 shows images and cameras of FH.

BikeCity2 (BC2) is generated by ray-tracking of a synthetic urban scene having real textures and by moving the camera along a trajectory that mimics that of BC1 (the "pose noises", i.e. relative poses between consecutive frames, are similar in both videos). Note that the cameras used to generate BC2 are rolling shutter like BC1 and WT. We obtain a video for each camera by compressing the output images using ffmpeg and options "-c:v libx264 -preset slow -crf 18". BC2 has a complete ground truth. Figure 3.11 also shows images of BC2.

There is also video WalkUniv (WU) using spherical camera - the Ricoh Theta S (Figure 3.13). This video is recorded while walking in the campus of the Université Clermont Auvergne.

Last CarCity (CC) is taken by the global shutter Ladybug 2 and has a similar trajectory as BC1. Moreover, it is mounted on a car (using a mast) and is about 4

FIGURE 3.11: Cameras (four Gopro Hero 3 in a cardboard) and images for BC1, WT and BC2. The rolling shutter always goes from right to left, the image motion goes toward left on the two left columns and goes toward right on the two right columns.

FIGURE 3.12: Cameras (four Gopro Hero 3 in their housings) and images for FH. The rolling shutter always goes from top to bottom, the image motion goes forward/toward right/backward/toward left.

FIGURE 3.13: Spherical camera (Ricoh Theta S) and images for WU. Both rolling shutter and image motion go from bottom to top.

meters above the ground, as shown in Figure 3.14. The camera records the suburban area with building, vegetation, parked cars. The images are uncompressed (all others are videos compressed using H.264).

FIGURE 3.14: Camera (PointGrey Ladybug 2) and images for CC. The cameras are global shutter. The camera pointing toward the sky is not used in our experiments.

Table 3.1 summarizes our main datasets. These datasets will be used in the next chapters.

Name (short name)	Camera	Type	f	r (mr)	b (cm)	l (m)	fr	FoV
BikeCity1 (BC1)	4*Gopro 3	RS	100	1.56	7.5	2500	50.4k	90
WalkTown (WT)	4*Gopro 3	RS	100	1.56	7.5	900	70.3k	90
FlyHill (FH)	4*Gopro 3	RS	48	1.06	18	1250	8.6k	90
BikeCity2 (BC2)	4*Gopro 3	RS	100	1.56	7.5	615	12.5k	90
CarCity (CC)	Ladybug 2	GS	15	1.90	6	2500	7.7k	72
WalkUniv (WU)	Theta S	RS	30	3.85	1.5	1260	29.4k	200

TABLE 3.1: Datasets: frame per second f, angular resolution r (milliradian), diameter b of multi-camera centers, approximate trajectory length l, numbers of frames fr and approximate horizontal FoV angle.

Chapter 4

Initialization of monocular self-calibration and synchronization

In this chapter, we present the two first stages in our self-calibration framework for omnidirectional multi-camera systems. Assuming that the monocular cameras are roughly equiangular and using an approximate knowledge of their FoV angle, Section 4.1 describes the initialization of two monocular camera models widely used in the bibliography: the polynomial distortion model [Lavest+98], [Sturm+11] and the unified camera model [Geyer+00], [Barreto06]. Using global shutter approximation, Section 4.2 introduces synchronization methods for two or more than two cameras. The methods presented here are published in [Lhuillier+15], [Nguyen+16b].

Contents

4.1 Monocular calibration initialization

The number of calibration methods has increased during recent years. These methods are based on different camera models. A survey of camera models used in the Computer Vision can be found in [Sturm+11]. The choice of a camera model depends on how appropriate the model is for the type of camera used. A good model balances goodness of fit with simplicity, i.e. it should explain the data well but not have too

many parameters. Roughly speaking, more complex models (with more parameters) are able to fit the data. But if more parameters than necessary are added then this makes up an overparameterization and *overfitting*. An overfitted model has poor predictive performance: it overacts to minor variation in the training data while the performance on a set of data not used for training becomes worse.

At first, we remind two simple geometric models: central perspective projection and equiangular projection for fisheye lens. Figure 4.1 shows a typical fisheye projection in comparison to the the central perspective projection. Let r_d be the radial distance in image (distance between image point and the center of distortion), f be the focal distance, μ be the incidence angle and α be the reflection angle. In the context of back-projection, μ is the angle spanned by the principal axis and the back-projected ray of an image point. The projection rays of the central perspective projection are straight lines and pass through a single point - the camera center (left of Figure 4.1). In contrast to the central perspective projection, a ray's incidence angle μ is different from its corresponding reflection angle α in fisheye projections (right of Figure 4.1). The equiangular projection is a type of fisheye projections in which the angles of incidence are translated linearly into radial distance. We have the following expression for the equiangular projection:

$$\mu = cr_d \tag{4.1}$$

where c is a constant.

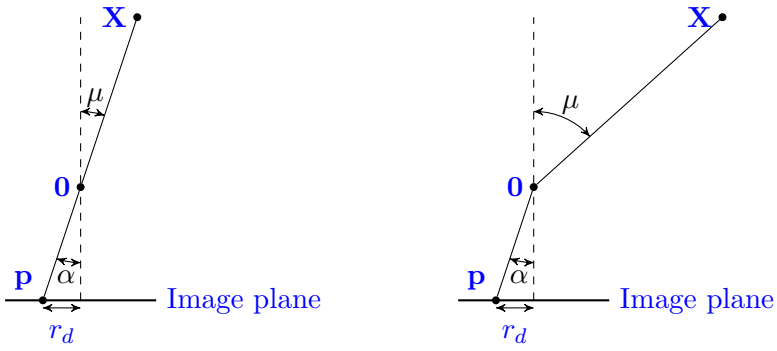


FIGURE 4.1: Central perspective geometry (left, $\alpha = \mu$) vs. fisheye projection geometry (right, $\alpha \neq \mu$).

Camera models involve matrices with particular properties that represent the camera mapping between the 3D world and a 2D image. We remind that the basic pinhole model (Section 2.1) is a linear relationship between image point position $\mathbf{p} = (u\ v)^\top$ and the direction of the associated camera ray $\tilde{\mathbf{X}}_\mathrm{C} = (X_\mathrm{C}\ Y_\mathrm{C}\ Z_\mathrm{C})^\top$ that can be expressed via the camera calibration matrix $\mathtt{K}$ in Eq.(2.2)

$$\begin{pmatrix} u \\ v \end{pmatrix} = \pi \left(\mathtt{K} \begin{pmatrix} X_\mathrm{C} \\ Y_\mathrm{C} \\ Z_\mathrm{C} \end{pmatrix} \right) \tag{4.2}$$

where

$$\mathtt{K} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } \pi \left(\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \right) = \begin{pmatrix} X/Z \\ Y/Z \end{pmatrix}. \tag{4.3}$$

Matrix K depends on four parameters, namely, focal parameters (f_x, f_y), principal point $\mathbf{p}_0 = (u_0, v_0)^\top$ and zero skew. In the following, we describe two monocular camera models that we use. The classical polynomial distortion model [Lavest+98], [Sturm+11], [Lébraly+11] is often used since its closed-form back-projection is useful for SfM tasks and epipolar geometry. It has several radial distortion parameters and can be applied to consumer camera like Gopro [GoPro]. The unified camera model [Geyer+00], [Barreto06] is also interesting since it deals with fisheye camera having FoV larger than 180^o (like those of spherical camera [ThetaS]) although it only has a single radial distortion parameter.

First, we review the two monocular camera models and explain how to initialize these camera models assuming that the cameras are roughly equiangular and using an approximate knowledge of their FoV angle in Subsection 4.1.1 and 4.1.2 (We remind that these approximations are suitable for an omnidirectional camera without privileged direction). Second, the approximate calibration is refined using monocular SfM and BA in Subsection 4.1.3. The experiments are in Subsection 4.1.4.

4.1.1 Polynomial distortion model

This model introduces a radial distortion function that represents the deflection of a ray from its perspective trajectory (see Figure 4.2). It models well for wide-angle distortion camera. We do not use tangential distortion in our work.

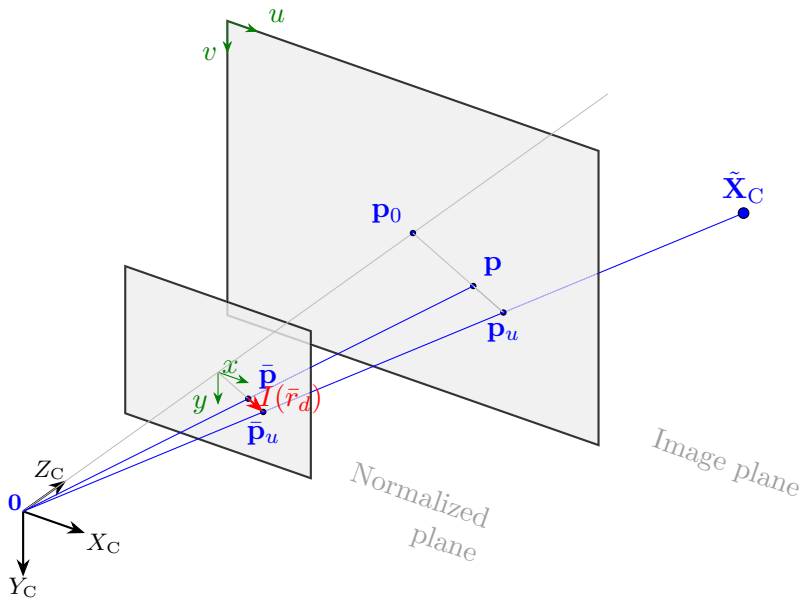


FIGURE 4.2: Polynomial distortion model.

4.1.1.1 Back-projection

The mapping from the distorted (original) image to the undistorted (rectified) image depends on radial distortion parameters k_i (tangential distortions are neglected). Let

$\mathbf{p}$ and $\mathbf{p}_u$ be the distorted and undistorted coordinates of a pixel, respectively. Their normalized coordinates $\bar{\mathbf{p}}$ and $\bar{\mathbf{p}}_u$ meet

$$\begin{pmatrix}\bar{\mathbf{p}}\\1\end{pmatrix} = \mathtt{K}^{-1}\begin{pmatrix}\mathbf{p}\\1\end{pmatrix} \text{ and } \begin{pmatrix}\bar{\mathbf{p}}_u\\1\end{pmatrix} = \mathtt{K}^{-1}\begin{pmatrix}\mathbf{p}_u\\1\end{pmatrix}. \tag{4.4}$$

Note that $\mathbf{p}_u$ is the inhomogeneous image coordinate of a point under ideal pinhole projection in Eq.(4.2). Let $\bar{r}_d = ||\bar{\mathbf{p}}||$ be the normalized radial distance in the distorted image. The distorted point coordinate is related to the undistorted point coordinate by a radial displacement. More precisely, the radial distortion is defined by

$$\bar{\mathbf{p}}_u = I(\bar{r}_d)\bar{\mathbf{p}} = \left(1 + \sum_{i=1}^{n} k_i \bar{r}_d^{2i}\right)\bar{\mathbf{p}}. \tag{4.5}$$

Lastly, the back-projected ray of pixel $\mathbf{p}$ has direction $(\bar{\mathbf{p}}_u^\top \ 1)^\top$ in the camera coordinate system. Figure 4.2 illustrates the back-projection using this model.

4.1.1.2 Forward projection

Here, we focus on the projection of 3D point $\tilde{\mathbf{X}}_\mathrm{C}$ in camera coordinates to 2D image point $\mathbf{p}$. Note that the back-projection is straightforward (closed form) thanks to Eqs.(4.4) and (4.5). But the forward projection is not (it is not closed form). First, we define vector $\boldsymbol{\theta}$ and functions g such that

$$\boldsymbol{\theta} = (f_x, f_y, \mathbf{p}_0, k_1, k_2, \ldots, k_n, \tilde{\mathbf{X}}_\mathrm{C}), \ \mathbf{p}_u = \pi(\mathtt{K}\tilde{\mathbf{X}}_\mathrm{C}), \tag{4.6}$$

$$\begin{pmatrix}u\\v\end{pmatrix} = \mathbf{z}, \ \begin{pmatrix}u_0\\v_0\end{pmatrix} = \mathbf{p}_0, \ \bar{r}^2 = \frac{(u-u_0)^2}{f_x^2} + \frac{(v-v_0)^2}{f_y^2}, \tag{4.7}$$

$$g(\mathbf{z}, \boldsymbol{\theta}) = \left(1 + \sum_{i=1}^{n} k_i \bar{r}^{2i}\right)(\mathbf{z} - \mathbf{p}_0) + \mathbf{p}_0 - \mathbf{p}_u \tag{4.8}$$

where $\mathtt{K}$ is defined in Eq.(4.3), $\begin{pmatrix}\bar{\mathbf{z}}\\1\end{pmatrix} = \mathtt{K}^{-1}\begin{pmatrix}\mathbf{z}\\1\end{pmatrix}$ and $\begin{pmatrix}\bar{\mathbf{p}}_u\\1\end{pmatrix} = \mathtt{K}^{-1}\begin{pmatrix}\mathbf{p}_u\\1\end{pmatrix}$.

Second, we show that $g(\mathbf{p}, \boldsymbol{\theta}) = \mathbf{0}$. $\mathbf{p} = \mathbf{z}$ implies that $\bar{r}_d$ in Subsection 4.1.1 and $\bar{r}$ are the same. We obtain $g(\mathbf{p}, \boldsymbol{\theta}) = \mathbf{0}$ by multiplying Eq.(4.5) on the left by $\begin{pmatrix}f_x & 0\\0 & f_y\end{pmatrix}$.

Last, using an approximate value $\tilde{\mathbf{p}}$ of $\mathbf{p}$, the projected point $\mathbf{p}$ is estimated by non-linear least-squares minimizing $\mathbf{z} \mapsto ||g(\mathbf{z}, \boldsymbol{\theta}_0)||^2$ where $\boldsymbol{\theta}_0$ is the current value of $\boldsymbol{\theta}$ (provided by initialization or previous iteration of BA). The implicit function Theorem implies that we locally have a $\mathcal{C}^1$ continuous function ψ such that $\mathbf{p} = \psi(\boldsymbol{\theta})$ if $\det \frac{\partial g}{\partial \mathbf{z}} \neq \mathbf{0}$.

4.1.1.3 Initialization

Here we present our equiangular initialization of $k_i, \mathbf{p}_0, f_x$ and f_y. The camera is equiangular if the angle μ between the principal axis $(0\ 0\ 1)^\top$ and the back-projected ray is proportional to the (non normalized) radial distortion r_d in the distorted image. We have $r_d = ||\mathbf{p} - \mathbf{p}_0||$. If the camera is equiangular, $f_x = f_y = f$ and there is a

constant c such that $\mu = cr_d$. Thus, $\mu = cf\bar{r}_d$. Since $\tan\mu = ||\bar{\mathbf{p}}_u||$ and using Eq.(4.5), we have

$$\tan(cf\bar{r}_d) = \tan\mu = \bar{r}_d + \sum_{i=1}^{n} k_i \bar{r}_d^{2i+1}. \tag{4.9}$$

Since tan is not polynomial, Eq.(4.9) can not be exact. We use Taylor's approximation

$$\tan\mu \approx \sum_{i=0}^{n} t_i \mu^{2i+1} = \mu + \frac{1}{3}\mu^3 + \frac{2}{5}\mu^5 + \frac{17}{315}\mu^7 + \frac{62}{2835}\mu^9 + \frac{1382}{155925}\mu^{11} + \dots \tag{4.10}$$

and identify coefficients between Eq.(4.9) and Eq.(4.10). We obtain $cf = 1$ using t_0 and $k_i = t_i$ if $i \geq 1$.

In practice, we initialize $\mathbf{p}_0$ at the image center and compute $f = r_d/\mu$ for a pixel $\mathbf{p}$ at the center of an image border where the half-FoV μ is approximately known.

4.1.2 Unified camera model

The unified camera model aims to model fisheye lenses with more than 180^o angle of view. It can model correctly projection of 3D points with zero and even negative Z_C. Only one projection parameter ξ is used to represent radial distortion. Overall we have 5 projection parameters: f_x, f_y, u_0, v_0, ξ.

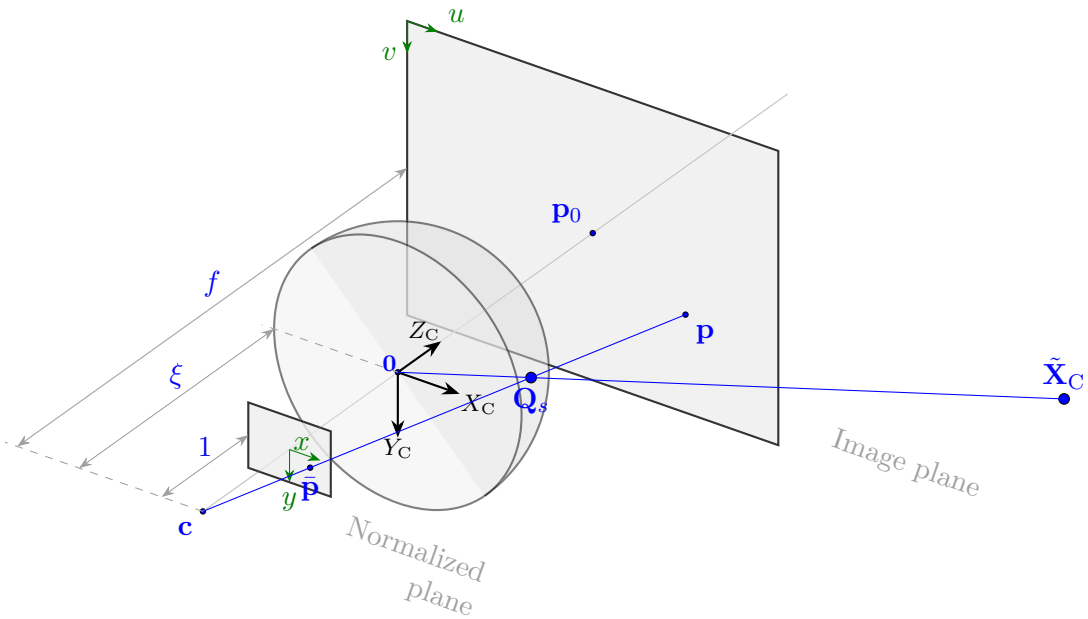


FIGURE 4.3: Unified camera model.

4.1.2.1 Forward projection

Figure 4.3 illustrates the forward projection using this model. Let $\mathcal{S}$ be the unit sphere in $\mathbb{R}^3$ centered at the center of camera coordinate system and let $\xi \in \mathbb{R}^+$. The projection $p(\tilde{\mathbf{X}}_C)$ of $\tilde{\mathbf{X}}_C$ by this model is obtained as follows:

1. $\tilde{\mathbf{X}}_{\mathrm{C}}$ is projected onto $\mathcal{S}$:

$$\mathbf{Q}_s = \frac{\tilde{\mathbf{X}}_{\mathrm{C}}}{||\tilde{\mathbf{X}}_{\mathrm{C}}||} = \frac{1}{\rho}\begin{pmatrix} X_{\mathrm{C}} \\ Y_{\mathrm{C}} \\ Z_{\mathrm{C}} \end{pmatrix} \tag{4.11}$$

   where $\rho = \sqrt{X_{\mathrm{C}}^2 + Y_{\mathrm{C}}^2 + Z_{\mathrm{C}}^2}$.

2. Perspective projection with center $(0, 0, -\xi)^\top$ of $\mathbf{Q}_s$ onto normalized plane:

$$\begin{pmatrix} \bar{\mathbf{p}} \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{X_{\mathrm{C}}}{Z_{\mathrm{C}} + \rho\xi} \\ \frac{Y_{\mathrm{C}}}{Z_{\mathrm{C}} + \rho\xi} \\ 1 \end{pmatrix} \sim \begin{pmatrix} \frac{X_{\mathrm{C}}}{\rho} \\ \frac{Y_{\mathrm{C}}}{\rho} \\ \frac{Z_{\mathrm{C}}}{\rho} + \xi \end{pmatrix} \tag{4.12}$$

3. Affine transformation by intrinsic parameter matrix $\mathtt{K}$ in order to obtain image point

$$\begin{pmatrix} \mathbf{p} \\ 1 \end{pmatrix} = \mathtt{K}\begin{pmatrix} \bar{\mathbf{p}} \\ 1 \end{pmatrix}. \tag{4.13}$$

   The projection can be rewritten as

$$p(\tilde{\mathbf{X}}_{\mathrm{C}}) = \pi\left(\mathtt{K}\left(\frac{\tilde{\mathbf{X}}_{\mathrm{C}}}{||\tilde{\mathbf{X}}_{\mathrm{C}}||} + \begin{pmatrix} 0 \\ 0 \\ \xi \end{pmatrix}\right)\right) \tag{4.14}$$

where $\mathtt{K}$ and π is defined by Eq.(4.3).

4.1.2.2 Back-projection

Here, we describe the mapping from a image point $\mathbf{p}$ to a point $\mathbf{Q}_s$ on the sphere (and its back-projected ray). The normalized coordinate of a pixel meets

$$\begin{pmatrix} \bar{\mathbf{p}} \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathtt{K}^{-1}\begin{pmatrix} \mathbf{p} \\ 1 \end{pmatrix}. \tag{4.15}$$

We know that a line joining the normalized point $\bar{\mathbf{p}}$ and the projection center $\mathbf{c}$ intersects the unit sphere at a point $\mathbf{Q}_s$. This point is defined as [Barreto03]

$$\mathbf{Q}_s = \begin{pmatrix} \eta x \\ \eta y \\ \eta - \xi \end{pmatrix} \tag{4.16}$$

where scale factor

$$\eta = \frac{\xi + \sqrt{1 + (x^2 + y^2)(1 - \xi^2)}}{x^2 + y^2 + 1}. \tag{4.17}$$

Demonstration. We choose η by using the fact that $\mathbf{Q}_s$ is on the unit sphere. Thus we have

$$||\mathbf{Q}_s|| = 1 \Leftrightarrow (x^2 + y^2 + 1)\eta^2 - 2\xi\eta + (\xi^2 - 1) = 0. \tag{4.18}$$

Let us consider the discriminant of this quadratic equation in η: $\Delta = 1+(x^2+y^2)(1-\xi^2)$. If $0 \leq \xi < 1$, Δ is always positive and Eq.(4.18) has two real roots, one positive and one negative. We choose the positive root. Now assume that $\xi > 1$. We have $\Delta \geq 0$, i.e. $\mathbf{Q}_s$ is defined, if

$$x^2 + y^2 \leq \frac{1}{\xi^2 - 1}. \tag{4.19}$$

In this case, Eq.(4.18) has two real positive roots. We choose the biggest one. In other words, we choose the intersection furthest from $\mathbf{c}$. In both cases, we obtain Eq.(4.17).

4.1.2.3 Initialization

Here we initialize $\xi, \mathbf{p}_0, f_x$ and f_y for an equiangular camera. Let μ be the angle between principal direction $(0\ 0\ 1)^\top$ and a back-projected ray, which is a half-line started at $\mathbf{0}$ with direction $\mathbf{Q}_s$. Appendix A shows that

$$f_x = f_y = f \Rightarrow \frac{||p(\tilde{\mathbf{X}}_\mathrm{C}) - \mathbf{p}_0||}{f} = \frac{\sin\mu}{\xi + \cos\mu}. \tag{4.20}$$

If the camera is equiangular, $f_x = f_y = f$ and there is a constant c such that $\mu = c||p(\tilde{\mathbf{X}}_\mathrm{C}) - \mathbf{p}_0||$. Since $\frac{\sin\mu}{\xi+\cos\mu}$ is not linear in μ, we approximate it thanks to Taylor expansions:

$$\begin{aligned} \frac{\sin\mu}{\xi + \cos\mu} \approx \frac{\mu - \mu^3/6}{\xi + 1 - \mu^2/2} &= \frac{\mu(1 - \mu^2/6)}{(1+\xi)(1 - \mu^2/(2\xi + 2))} \\ &= \frac{\mu}{1+\xi}\left(1 + \mu^2\left(\frac{1}{2\xi+2} - \frac{1}{6}\right) + \mathcal{O}(\mu^4)\right). \end{aligned} \tag{4.21}$$

We initialize $\xi = 2$ such that this approximation is linear in μ. Now we distinguish two cases for the initialization of $\mathbf{p}_0$ and f. If every pixel of the (rectangular) image has a back-projected ray, we initialize $\mathbf{p}_0$ at the image center and take point $\mathbf{p}_1$ at the center of an image border where the half-FoV μ is approximately known. Otherwise, we assume that the pixels that have back-projected rays form a disk whose radius and center can be estimated. Then we initialize $\mathbf{p}_0$ by the disk center and take point $\mathbf{p}_1$ at the disk boundary where the half-FoV μ is approximately known. In both cases, we should choose $\mu \leq 2\pi/3$ according to Appendix A and f is initialized by Eq.(4.20) using $p(\tilde{\mathbf{X}}_\mathrm{C}) = \mathbf{p}_1$.

4.1.3 Monocular calibration refinement

The monocular camera model is initialized as in the previous sections assuming that the camera is roughly equiangular and using an approximate knowledge of their FoV angles. We apply monocular SfM summarized in Section 2.4 (global shutter assumption). The camera poses $(\mathtt{R}_i, \mathbf{t}_i)$, 3D points $\mathbf{X}_j$ and intrinsic parameters vector $\mathbf{I}$ are refined by a LM algorithm (see Section 2.3) which minimizes the reprojection errors. The residual vector is described as

$$\boldsymbol{\epsilon}_{ij} = \mathbf{p}(\mathbf{I}, \mathtt{R}_i, \mathbf{t}_i, \mathbf{X}_j) - \tilde{\mathbf{p}}_{ij} \tag{4.22}$$

where $\tilde{\mathbf{p}}_{ij}$ is an inlier 2D observation of the 3D point $\mathbf{X}_j$ in the i-th keyframe and $\mathbf{p}$ is the projected point of $\mathbf{X}_j$ in the i-th keyframe. We remind that $\tilde{\mathbf{p}}_{ij}$ is considered as an inlier if $||\boldsymbol{\epsilon}_{ij}||$ is less than a threshold (4 pixels in our experiments). The minimized cost function is

$$F = \frac{1}{2} \sum \boldsymbol{\epsilon}_{ij}^{\top} \boldsymbol{\epsilon}_{ij}. \tag{4.23}$$

Under the standard assumption that the image noise due to point detection follows zero-mean normalized identical and independent Gaussian vectors, the result obtained by BA is a Maximum-Likelihood Estimation (this assumption is not true in the undistorted space, especially in the case of large distortions between undistorted and distorted spaces).

4.1.4 Experiments

As a reminder, the monocular camera model is initialized assuming that the cameras are roughly equiangular and using an approximated knowledge of their FoV angle. The polynomial distortion model is used for the wide-angle cameras: Point Grey Ladybug2 and GoPro. The unified camera model is used for fisheye camera like Ricoh Theta S that has FoV larger than 180^o. Here, we apply monocular SfM on the 2000 first images in our experiments and calibration refinement by BA for every camera. This is enough for initializing the next steps of our approach including synchronization.

In this section, we show results for monocular cameras. The cameras and the videos are summarized in Table 3.1. Figure 4.4(a) shows images for CC-cam2 using Ladybug2 camera. The camera is mounted on a car and records the video in an average traffic, so there are moving cars and pedestrians. Figure 4.5(a) shows images for WT-cam1 using GoPro camera mounted on a helmet. This video is recorded while the user is walking in the streets of a city. Images acquired by Theta S camera include two fisheye sub-images. This camera is also fixed on a helmet and records video sequence while walking in the campus of the Université Clermont Auvergne (UCA) (see Figure 4.6(a) for WU-cam1).

For monocular wide-angle camera, the intrinsic parameters are initialized with approximation FoV = 72^o ($360^o/5$) for CC-cam2 or 90^o ($360^o/4$) for WT-cam1 and equiangular assumption (see Subsection 4.1.1). For fisheye camera, as explained in Subsection 4.1.2, we initialize $\xi = 2$ and FoV = 200^o. We apply monocular SfM (Section 2.4) and obtain: 255 keyframes and 24876 3D points for monocular CC-cam2; 63 keyframes and 3989 3D points for WT-cam1; and 133 keyframes and 32231 3D points for WU-cam1. Table 4.1 provides the numerical values of intrinsic parameters. We check that the BA quantitatively improves the intrinsic parameters, especially f_x, f_y, k_1 and k_2 (when ground truth is known in Table 4.1). Furthermore, Figures 4.4(b), 4.5(b) and 4.6(b) show a top view of reconstruction before and after BA. The BA qualitatively improves the camera trajectories shown in the figures. We observe that the BA corrects scale drift for CC and WT; and corrects rotation drift for WU.

However, there is a limitation. SfM can fail for a video due to the combination of two difficulties: lack of texture and approximate calibration. We assume that there is at least one textured enough video such that this is successful. Since the cameras have the same configuration (this assumption is suitable for an omnidirectional camera without a privileged direction), we benefit by the refined intrinsic parameters by BA to redo the monocular SfM of the other videos. Thus, a difficulty is reduced for the less textured video and the risk of failure decreases.

(a) Sequence with 1024×768 resolution at 15 frames/second

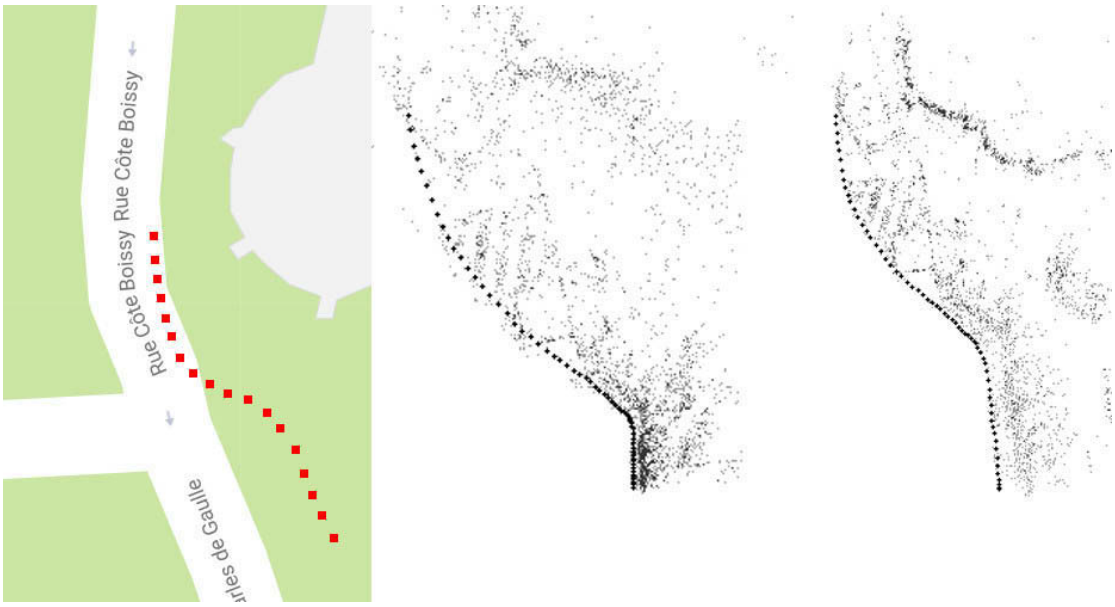
(b) Left: Map and trajectory; Center: Before BA; Right: After BA

FIGURE 4.4: Top view of monocular Point Grey camera (CC-cam2) reconstruction.

	CC-cam2			WT-cam1			WU-cam1		
c	pat_2	72	$72r$	pat_1	90	$90r$		200	$200r$
f_x	552.777	611.155	558.169	581.133	611.155	582.541	f_x	890.172	859.494
f_y	552.909	611.155	555.104	581.182	611.155	578.819	f_y	890.172	861.830
u_0	511.597	512.0	515.268	639.420	640	640.295	u_0	480.0	478.510
v_0	383.882	384.0	390.146	481.097	480	478.127	v_0	480.0	479.187
k_1	0.417	0.333	0.417	0.369	0.333	0.367			
k_2	0.093	0.400	0.066	0.067	0.400	0.056			
k_3	0.420	0.054	0.578	0.004	0.054	0.029	ξ	2.000	2.023
k_4	−0.376	0.022	−0.635	0.017	0.022	−0.007			
k_5	0.311	0.008	0.467	0.007	0.008	0.015			

TABLE 4.1: Intrinsic calibration parameters. Notation: *pat* - ground truth (obtained with a calibration pattern [Lavest+98]), 72/90/200 - initial calibration and $72r/90r/200r$ - refined calibration.

(a) Sequence with 1280 × 960 resolution at 100 Hz



(b) Left: Map and trajectory; Center: Before BA; Right: After BA

FIGURE 4.5: Top view of monocular Gopro camera (WT-cam1) reconstruction.

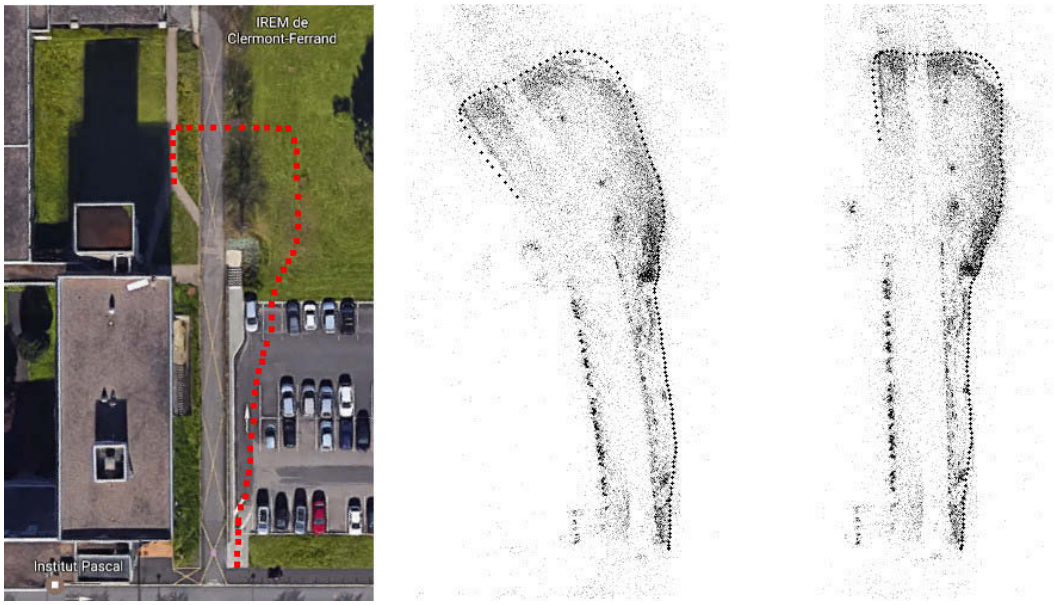
4.2 Synchronization

4.2.1 Motivation

There are several ways to synchronize cameras. Audio-based synchronization is possible if a distinct sound is available and the cameras do not have audio/video synchronization issues. For example, our GoPro Hero3 camera provides poor audio/video synchronization (according to the user guide of VideoStitch (http://www.video-stitch.com/)). In many cases like our DIY multi-camera based on GoPro, the manufacturer provides a wifi-based synchronization. However, the resulting time offsets between videos are too inaccurate for applications: about 0.04s and sometimes above 0.1s in our experiments. Assume that a central multi-camera moves at 20km/h (e.g. biking in a city) and two cameras have a time offset equal to only 0.02s, then explain consequences on a 360 video obtained by video stitching. If we neglect this offset, the two videos are stitched as if they have same camera centers at same frame number, although the distance between these centers is 0.11m (20/3.6*0.02). This generates artifacts in the 360 video due to foreground objects that are in the FoV shared by the two cameras.

In our context, applications like 360 video need a central multi-cameras. However, the baseline defined by the distance between the centers of two cameras is not zero. The

(a) Sequence with 960×960 resolution at 30 frames/second



(b) Left: Map and trajectory (best viewed in the electronic version); Center: Before BA; Right: After BA

FIGURE 4.6: Top view of monocular fisheye camera (WU-cam1) reconstruction.

smaller the baseline, the better the stitching quality. The user/manufacturer can reduce the baseline thanks to a small number of cameras. Since a small number of cameras also reduces the shared FoV (see Figure 1.1), the synchronization methods based on correspondences between the features observed in two videos are not recommended. But, we can benefit an important constraint: the cameras are moving jointly. The larger motion in one video, the larger motion in the others. A method close to ours is proposed in [Spencer+04]: transformations (affine, homography) are estimated between consecutive frames of every video. The estimated time offset is the one that best "compares" the transformations between two videos. However, the transformations in [Spencer+04] are heuristic (translation) or uncalibrated (homography) ignoring radial distortion. Here, we propose a method based on instantaneous angular velocity estimated by monocular SfM, which does not have the inconveniences above.

We remind that all cameras have the same setting for an omnidirectional multi-camera without privileged direction. So the ratio of frames rate of the two sequences is equal to 1. Let f be the (same) frequency of the cameras. Let s_i be the number of skipped frames at the beginning of the i-th video such that the sequels of the videos are frame-accurately (FA) synchronized. Let Δ_j be the subframe time offset between the j-th video and the first video (the 0-th video). It means that if we shift s_0/f seconds in the first video and shift $s_j/f + \Delta_j$ seconds in the j-th video then two videos are synchronized correctly. Figure 4.7 shows all notations.

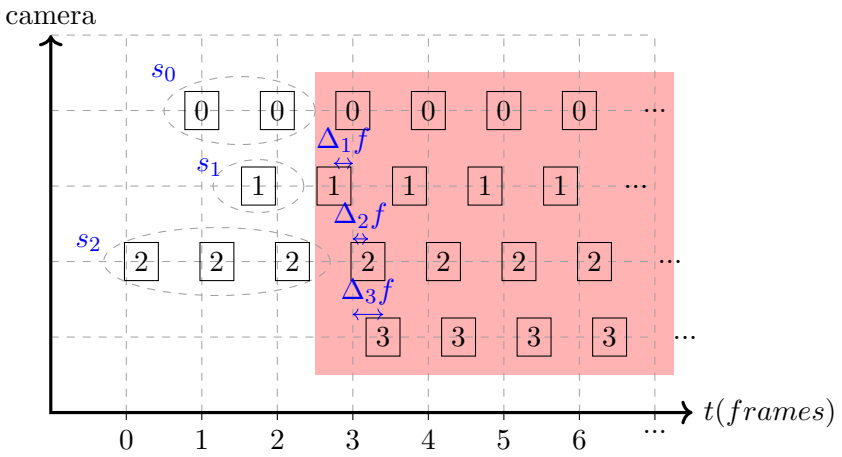


FIGURE 4.7: Synchronization problem. Note that s_i is a number of frames, Δ_j is in seconds.

The synchronization initialization is required by the multi-camera SfM-BA and has two steps. First, Subsection 4.2.2 estimates instantaneous angular velocities (IAV) thanks to monocular SfM-BA and GS approximation. Then time offsets are computed by correlation of IAVs of different cameras; Subsection 4.2.3 and 4.2.4 describe the two- and multi-camera cases respectively.

4.2.2 Instantaneous angular velocity (IAV)

We assume that every monocular video is reconstructed such that every frame has a computed pose (both keyframes and non-keyframes). Thus keyframe-based SfM like [Mouragnon+09] should be followed by pose calculations for the non-keyframes and by BA. In practice, it is sufficient to reconstruct few thousands of frames at the video beginning for the synchronization initialization (we use 2k frames as in Section 4.1).

Let $\mathtt{R}_i^t$ be the rotation of the pose of the t-th frame in the reconstruction of the i-th video. The IAV θ_i^t at the t-th frame (of the i-th video) is approximated by the angle of rotation $\mathtt{R}_i^{t+1}(\mathtt{R}_i^t)^\top$, i.e.

$$\theta_i^t = \arccos\left(\frac{\operatorname{trace}(\mathtt{R}_i^{t+1}(\mathtt{R}_i^t)^\top) - 1)}{2}\right). \tag{4.24}$$

We omit the FpS coefficient since all cameras have the same. Intuitively, the IAV is the same for two frames of different but jointly moving cameras if they are taken at the same time. Formerly, we show this as follows. We remind properties of change of basis in $\mathbb{R}^3$ expressed by rotation matrices: the columns of $\mathtt{R}_{A,B}$ are vectors of B expressed using coordinates in A, we have $\mathtt{R}_{A,B}^\top = \mathtt{R}_{B,A}$ and $\mathtt{R}_{A,B} = \mathtt{R}_{A,C}\mathtt{R}_{C,B}$.

Since the monocular SfMs are not done in the same coordinate system, the notation $\mathtt{R}_i^t$ in Eq.(4.24) is ambiguous. Here, we write instead $\mathtt{R}_{w_i,i}^t$ where w_i is the (world) basis of the i-th video reconstruction (vectors of the i-th camera at frame t expressed using coordinates in w_i). Furthermore, we note that $\mathtt{R}_{w_i,w_j}$ and $\mathtt{R}_{j,i}$ do not depend on frame numbers (the former is obvious, the latter is due to the fact that the cameras are rigidly mounted). If the t_i-th frame of the i-th camera and the t_j-th frame of the j-th camera

are taken at the same time,

$$\mathtt{R}_{w_j,w_i}\mathtt{R}^{t_i}_{w_i,i} = \mathtt{R}^{t_j}_{w_j,j}\mathtt{R}_{j,i} \tag{4.25}$$

Since the cameras have the same FpS, we also have

$$\mathtt{R}_{w_j,w_i}\mathtt{R}^{t_i+1}_{w_i,i} = \mathtt{R}^{t_j+1}_{w_j,j}\mathtt{R}_{j,i} \tag{4.26}$$

Thanks to Eq.(4.25) and Eq.(4.26), we obtain

$$\mathtt{R}^{t_j+1}_{w_j,j}(\mathtt{R}^{t_j}_{w_j,j})^\top = \mathtt{R}_{w_j,w_i}\mathtt{R}^{t_i+1}_{w_i,i}(\mathtt{R}^{t_i}_{w_i,i})^\top\mathtt{R}^\top_{w_j,w_i}. \tag{4.27}$$

Since $\text{trace}(\mathtt{XY}) = \text{trace}(\mathtt{YX})$, we obtain

$$\text{trace}\left(\mathtt{R}^{t_j+1}_{w_j,j}(\mathtt{R}^{t_j}_{w_j,j})^\top\right) = \text{trace}\left(\mathtt{R}^{t_i+1}_{w_i,i}(\mathtt{R}^{t_i}_{w_i,i})^\top\right). \tag{4.28}$$

Thus $\theta^{t_j}_j = \theta^{t_i}_i$ according to Eq.(4.24).

4.2.3 Synchronize two cameras

We compute the IAV table for every camera and find the time offset that maximizes the correlation (ZNCC) between two such tables (matches two subtables with the same length in different tables). In more details, let $o_{i,j}$ be the frame offset between the i-th and j-th cameras. According to Subsection 4.2.2, we have $\theta^t_i \approx \theta^{t+o_{i,j}}_j$. The frame offset $o_{i,j}$ maximizes correlation $\text{ZNCC}_{i,j}$ between vectors θ_i and θ_j:

$$\text{ZNCC}_{i,j}(o_{i,j}) = \frac{\sum_{t\in T}\left((\theta^t_i - m_i)(\theta^{t+o_{i,j}}_j - m_j)\right)}{\sqrt{\sum_{t\in T}(\theta^t_i - m_i)^2}\sqrt{\sum_{t\in T}(\theta^{t+o_{i,j}}_j - m_j)^2}} \tag{4.29}$$

where $T \subset \mathbb{N}$ is a set of consecutive frame number and m_i and m_j are the means of the vector θ_i and θ_j, respectively:

$$m_i = \frac{1}{|T|}\sum_{t\in T}\theta^t_i \quad \text{and} \quad m_j = \frac{1}{|T|}\sum_{t\in T}\theta^{t+o_{i,j}}_j. \tag{4.30}$$

We also introduce a simple subframe accurate (SFA) refinement method. The subframe offsets are estimated like sub-pixelic disparity using a quadratic fit: first approximate the mapping from $o_{i,j}$ to $\text{ZNCC}_{i,j}$ using a quadratic polynomial defined by its 3 values at $o_{i,j}+\{-1,0,1\}$; then estimate $\epsilon_{i,j}$ such that $o_{i,j}+\epsilon_{i,j}$ maximizes this polynomial. Hence,

$$\epsilon_{i,j} = (\Delta_j - \Delta_i)f. \tag{4.31}$$

In contrast to the FA offsets $o_{i,j} \in \mathbb{Z}$, the SFA offsets $o_{i,j} + \epsilon_{i,j} \in \mathbb{R}$ and are not used for the input of our BA in the next chapters. This is an alternative SFA synchronization method assuming that the cameras are global shutter.

4.2.4 Consistently synchronize more than 2 cameras

We remind that the goal of the FA synchronization is to skip s_i frames at the beginning of the i-th video such that the sequels of the videos are FA synchronized (This is required for multi-camera SfM). Thus $o_{i,j} = s_j - s_i$ for all $i \neq j$, which in turn imply that the sum of offsets along every loop in the camera graph should be zero (e.g. we should have $o_{0,1} + o_{1,2} + o_{2,0} = 0$ for loop $0 \to 1 \to 2 \to 0$). However, such a sum can be non-zero since the offsets are estimated independently.

There are several ways to deal with this loop constraint. First only compute offsets $o_{0,i}$. But this solution privileges the first camera. Second compute all $o_{i,j}$, generate candidate offsets around $o_{i,j}$ for every pair (i, j), and select the candidate offsets that maximizes $\sum_{i \neq j} \text{ZNCC}_{i,j}$ such that the sum of candidate offsets along every loop is zero. We implement an intermediate and simple solution where every camera has the same importance assuming that the cameras are symmetrically mounted around a symmetry axis: we only consider the spatial adjacency of the n cameras, i.e. we only compute offsets $o_{0,1}$, $o_{1,2}$, $\ldots, o_{n-2,n-1}$, $o_{n-1,0}$ (instead of all $o_{i,j}$) and only use loop $0 \to 1 \to \ldots \to n-1 \to 0$ (instead of all loops) in the scheme above. In practice, we found that it is sufficient to choose candidate offsets that differ from the initial ones by +1 or 0 or -1.

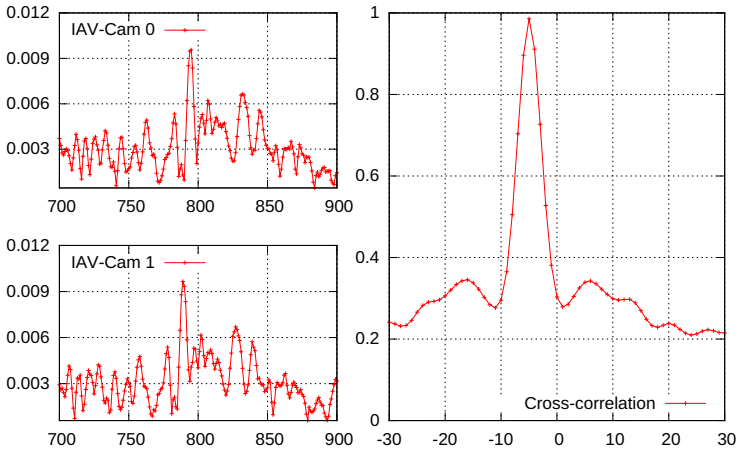
4.2.5 Experiments

In Subsection 4.2.5.1, we experiment our synchronization method using monocular SfM for every camera at the beginning of the datasets summarized in Table 3.1 with the initialization of intrinsic parameters in Section 4.1. We also show the robustness of our method with respect to intrinsic parameters and compare to the other methods in the literature (Subsection 4.2.5.2).

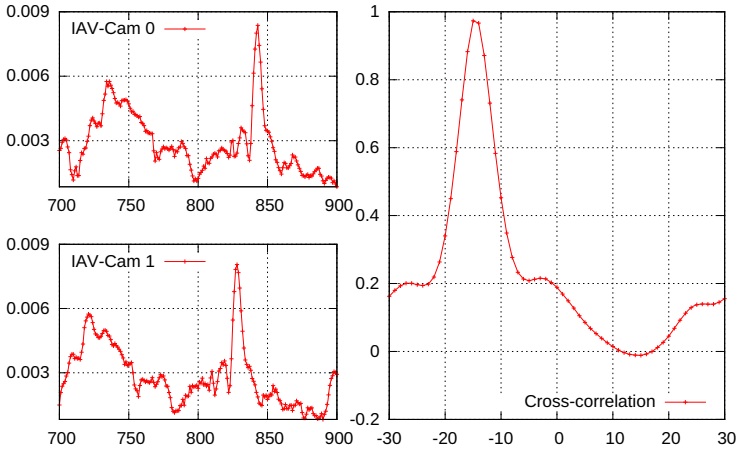
4.2.5.1 Frame-accurate synchronization

Using the initialization of intrinsic parameters in Section 4.1 (equiangular approximation and BA refinement), Figure 4.8 draws the IAV in Eq.(4.24) (in radian) for consecutive frames taken in a rectilinear segment of the trajectory and the correlation function in Eq.(4.29) (that maps FA offset candidate $o_{0,1}$ to $\text{ZNCC}_{0,1}$) for cameras 0 and 1. This is done for biking (BC1), walking (WT) and car+mast (CC). There are similar variations of the IAV for different cameras and a single maximum of the ZNCC except for WT. Two consecutive offsets of WT have very similar greatest ZNCC values and the other ZNCC values are below, which suggest a half-frame residual time offset. These examples can convince the reader that we have enough information in the IAV to obtain a FA synchronization (at least if the cameras are helmet-held or mounted on a car thanks to a mast).

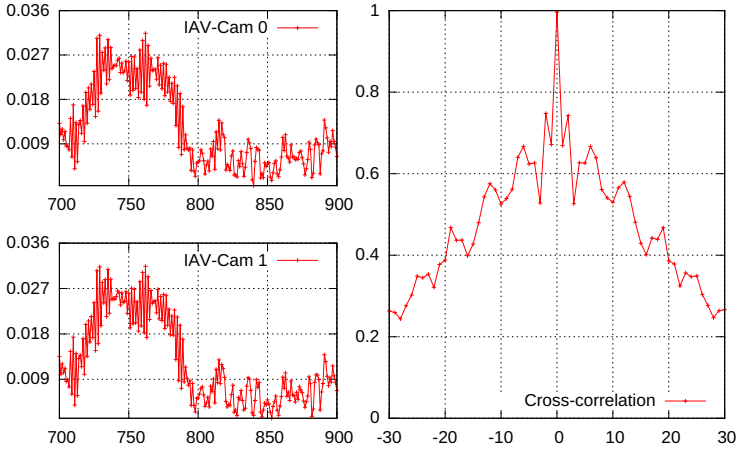
Table 4.2 shows the FA time offsets with loop constraint for all sequences. First we examine $o_{i,i+1}$ for the four Gopro Hero 3 cameras in BC1, WT and FH. Different experiments have different $o_{i,i+1}$ although they are taken by the same cameras. Thus synchronization should be done at every experiment. Furthermore, the wifi-based synchronization of the Gopro is not very accurate: about 0.04s and sometimes above 0.1s (reminder: their FpS is 100Hz for BC1 and WT; or 48Hz for FH). Second we check that the $o_{i,i+1}$ are FA when their ground truths are known (BC2, CC and WU). Using



(a) BC1



(b) WT



(c) CC

FIGURE 4.8: IAV for two cameras (left) and their cross-correlation curves (right) for rectilinear trajectory segments of trajectories. In the left, we have frame numbers (x-axes) and IAVs in radian (y-axes). In the right, we have offset candidates $o_{i,j} \in \mathbb{Z}$ (x-axes) and its ZNCC correlation in $[-1, 1]$ (y-axes).

these results, we skip few frames in each video such that the sequences of videos are FA synchronized: frames with the same index are taken at the same time up to the inverse of FpS.

Name	$o_{0,1}$	$o_{1,2}$	$o_{2,3}$	$o_{3,4}$	ZNCC_1	ZNCC_2
BC1	-5	3	4	na	3.912	3.884
WT	-15	-1	14	na	3.919	3.918
FH	-1	1	-2	na	3.991	3.983
BC2	0	0	0	na	3.915	3.907
CC	0	0	0	0	4.987	4.347
WU	0	na	na	na	0.993	0.677

TABLE 4.2: FA time offsets $o_{i,i+1}$ with loop constraint for all videos. ZNCC_1 is the greatest sum of the n computed time offsets and ZNCC_2 is the second greatest ZNCC ($-n \leq \text{ZNCC} \leq +n$). We remind that $o_{i,i+1}$ counts a signed number of frames between the i-th and the $i+1$-th videos (Since the multi-camera has n cameras numbered i from 0 to $n-1$, $o_{i,i+1}$ is "na" if $i+1 \geq n$).

4.2.5.2 Robustness and comparisons

We detail the experiments on WT sequence with respect to several calibration inputs and compare our synchronization method to the other methods: an audio-based method (like http://www.video-stitch.com/) and a translation-based method [Spencer+04].

We introduce some notations. Let A_c be the name of our rotation-based (IAV) method using calibration c. If $c = pat$, the cameras are calibrated using a planar calibration pattern [Lavest+98]. If $c = 90$, the intrinsic parameters are such that the cameras are equiangular and their FoV in the horizontal direction is approximately known (90^o for the WT case). If $c = 90r$, first we apply SfM using $c = 90$ on keyframes of the video; then refine c using a global BA; lastly, use SfM once again for all frames with the refined c. Table 4.1 presents calibration results for $c = \{pat, 90, 90r\}$. Let T_f be the name of the translation-based method such that the mean of translation moduli is computed in the complete frame of a camera. Let T_b be a variant of T_f: the mean is computed in a small area including the common FoV with the other cameras (most left and right columns of images in Figure 3.11). We expect T_b to be better than T_f if the translation magnitude evolves similarly in different cameras, especially in the common FoV. Let S be the sound-based synchronization method using correlation of sound (sound replaces IAV/2D translation moduli in the tables). For WT, there are four offsets $o_{0,1}, o_{1,2}, o_{2,3}$ and $o_{3,0}$ between adjacent cameras that are computed by one of the six methods above. Let $L = o_{0,1} + o_{1,2} + o_{2,3} + o_{3,0}$. Since we have a loop of camera neighbors (Figure 3.11), L should be 0 by enforcing a loop constraint as in Subsection 4.2.4. We distinguish L_i, the sum of integer offsets $o_{i,i+1}$ (FA), and L_r the sum of real offsets $o_{i,i+1} + \epsilon_{i,i+1}$ (SFA) (Subsection 4.2.3).

The estimated offsets without and with loop constraint are showed in Table 4.3 and Table 4.4, respectively. We see that the offsets computed by A_c are roughly the same for the attempted calibration c. The better the c value, the better (smaller) the $|L_r|$ in Table 4.3 and the better (larger) the ZNCC scores in Table 4.4. In comparison, the $|L_r|$ and ZNCC scores of methods S, T_f and T_b are the worst. The results of T_f and T_b are not surprising since the translation-based methods are heuristic. One reason for the

results of the sound-based method S is that some cameras, including our GoPro Hero3, provide poor audio/video synchronization.

A_c	$o_{0,1}$	$o_{1,2}$	$o_{2,3}$	$o_{3,0}$	L_i	L_r
A_{90}	-15	-1	14	1	-1	-0.023
A_{90r}	-15	-1	14	1	-1	-0.022
A_{pat}	-15	-1	14	1	-1	-0.020
T_f	-12	0	12	0	0	0.100
T_b	-15	0	14	0	-1	-1.450
S	-18	-2	17	2	-1	-0.460

TABLE 4.3: Time offsets without loop constraint for WT. L_i and L_r are the sum of four integer and real time offsets, respectively.

A_c	$o_{0,1}$	$o_{1,2}$	$o_{2,3}$	$o_{3,0}$	ZNCC_1	ZNCC_2
A_{90}	-14	-1	14	1	3.708	3.707
A_{90r}	-15	-1	14	2	3.919	3.918
A_{pat}	-15	-1	14	2	3.915	3.915
T_f	-12	0	12	0	2.610	2.421
T_b	-15	0	14	1	3.319	3.221
S	-18	-2	18	2	1.680	1.630

TABLE 4.4: Time offsets with loop constraint for WT. ZNCC_1 is the largest sum of the four ZNCCs of the four computed time offsets, ZNCC_2 is the second largest ZNCC. $-4 \leq \text{ZNCC} \leq 4$.

4.3 Conclusion

This chapter provides an estimate of the monocular intrinsic parameters and the synchronization between the videos (which will be refined later in our process) using the global shutter approximation on our cameras. First, we initialize the intrinsic parameters of the monocular cameras using assumptions that are suitable to an omnidirectional multi-camera without privileged directions: all cameras have the same setting and are roughly equiangular with an approximately known FoV angle. Second, we apply to a video a standard monocular SfM followed by BA that refines the intrinsic parameters. Since the cameras have the same setting, we use these intrinsic parameters to apply SfM to the other videos. Now every frame in every monocular video has a pose (in practice we only need few thousands of frames at the video beginning). Third, we propose to synchronize the cameras as follows: estimate the instantaneous angular velocities (IAV) from the poses, then compute the time offsets by correlation of IAVs of different cameras. Note that the initialization of intrinsic parameters is not the core of the thesis and can be improved by several ways: monocular self-calibration using previous methods, automatic choice of the first video (e.g. take the most textured one to reduce the risk of SfM failure), selection of a video segment where SfM can be applied safely (avoid pure rotation motion and low textured/blurred frames).

Chapter 5

Multi-camera bundle adjustment assuming global shutter and frame-accurate synchronization

Cameras are used widely in mobile robotic, human-driven vehicles and 3D modeling; and a wide FoV is usually desirable for accurate ego-motion estimation and complete 3D reconstruction as well as navigation and localization. While this can be achieved by using single catadioptric omnidirectional camera, this is also provided by omnidirectional multi-cameras. Multi-camera systems such as these are investigated in our work. A good accuracy for both extrinsic and intrinsic calibration is usually required.

In this chapter, we improve the previous method [Lébraly+11] that aims at calibration of non-overlapping rigidly linked multi-camera rig without requirement for calibration patterns or other infrastructure. The multi-camera system moves in a natural static scene. The BA designed for multi-camera [Lébraly+11] refines the relative poses between the cameras, the poses of multi-camera frame and 3D points by minimizing a reprojection error. But the intrinsic parameters are not simultaneously refined and the reprojection error is in the undistorted (i.e. rectified) space of the classical polynomial distortion model (see Section 4.1.1). We improve this BA in two points: refine intrinsic parameters (not only the extrinsic parameters and 3D geometry) and minimize the reprojection error in the right space - the original space where the image points are detected. The extension for the unified camera model (see Section 4.1.2) is straightforward. In this chapter, we assume that the cameras are global shutter and frame-accurately synchronized thanks to the method in Chapter 4. The method presented here is published in [Lhuillier+15], [Nguyen+16b].

Contents

5.1 Notations

We consider $n \geq 2$ jointly moving cameras with small or even empty shared FoV. The multi-camera system is moving while each camera observes a static scene $\mathcal{X}$ (cloud of 3D points), and $m+1$ poses of the multi-camera system are obtained by a multi-camera SfM. Let $\mathtt{T}^i_{\mathrm{M}}$ be the homogeneous transformation of the i-th multi-camera pose in the world coordinate system ($i \in [0, m]$), $\mathtt{T}^j_{\mathrm{C}}$ be the homogeneous transformation of the j-th camera in the multi-camera coordinate system ($j \in [0, n \quad 1]$), see Figure 5.1. Each homogeneous transformation $\mathtt{T}$ is represented with a rotation $\mathtt{R}$ and a translation $\mathbf{t}$ such that

$$\mathtt{T}^i_{\mathrm{M}} = \begin{bmatrix} \mathtt{R}^i_{\mathrm{M}} & \mathbf{t}^i_{\mathrm{M}} \\ \mathbf{0}^\top & 1 \end{bmatrix} \text{ and } \mathtt{T}^j_{\mathrm{C}} = \begin{bmatrix} \mathtt{R}^j_{\mathrm{C}} & \mathbf{t}^j_{\mathrm{C}} \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{5.1}$$

where $(\mathtt{R}^i_{\mathrm{M}}, \mathbf{t}^i_{\mathrm{M}})$ denote the rotation and translation of the i-th multi-camera pose in the world-coordinate system, $(\mathtt{R}^j_{\mathrm{C}}, \mathbf{t}^j_{\mathrm{C}})$ denote the relative rotation and translation of the j-th camera in the multi-camera coordinate system. The extrinsic parameters $\mathtt{T}^j_{\mathrm{C}}$ are assumed to be constant over time.

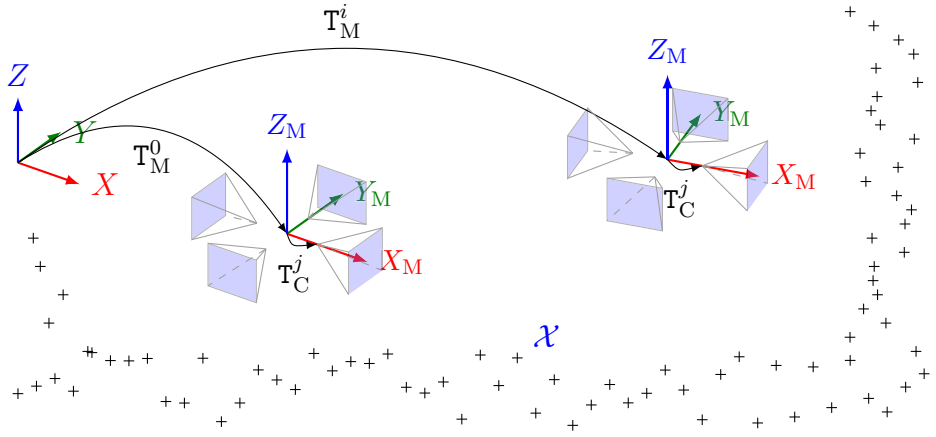


FIGURE 5.1: Multi-camera rig moving along a static scene.

Figure 5.2 illustrates our proposed pipeline for multi-camera calibration. First, initial intrinsic parameters $\mathbf{I}^j$ and the extrinsic parameters $\mathtt{T}^j_{\mathrm{C}}$ are initialized in Section 5.2. Second, the calibration process requires to compute the keyframe multi-camera poses $\mathtt{T}^i_{\mathrm{M}}$ and 3D structure (see SfM algorithm in Section 2.4). Finally, these parameters will be refined all together using the multi-camera bundle adjustment (MCBA) in Section 5.3 and Section 5.4.

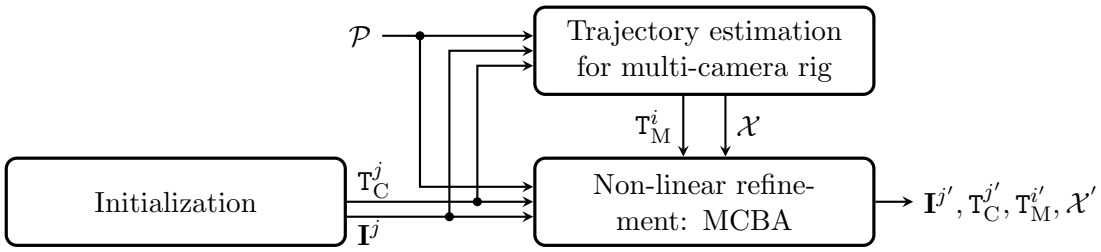


FIGURE 5.2: Calibration scheme of multi-camera system. $\mathcal{P}$ is a set of observations in selected key frames. Apostrophe ($'$) indicates refined parameters.

5.2 Initialization

First, we remind that the intrinsic parameters are initialized assuming that the monocular cameras are GS and roughly equiangular with an approximately known FoV. The cameras have the same setting: frequency, image resolution and FoV. Our assumptions are easy to meet in practice and are suitable for an omnidirectional camera without a privileged direction. Section 4.1 provides more details about the initialization of the intrinsic parameters.

Second, we also remind that a FA synchronization between all videos is obtained using the method based on IAV estimated by monocular SfM (see Section 4.2). The monocular videos are synchronized by removing few frames at their beginning, i.e. the videos are synchronized up to the inverse of FpS. We define the i-th frame of the multi-camera by a concatenation of sub-images, every of them is the i-th frame of a monocular camera. From now on, we use word *frame* for "frame of the multi-camera" and the *video* is the sequence defined by all these frames.

Third, while the relative transformation between cameras can be initialized linearly using the methods mentioned in Section 3.3, we initialize them using the central approximation for translation and an approximately known inter-camera rotations: n cameras are symmetrically mounted around a symmetry axis $\mathbf{a}$. This is enough to feed the bundle adjustment in our case. More precisely, the central approximation means

$$\forall j \in \{0, \dots, n-1\}, \mathbf{t}^j_{\mathrm{C}} = \mathbf{0}. \tag{5.2}$$

Let $\mathcal{R}(\mathbf{a}, \theta)$ be the rotation around axis $\mathbf{a}$ with angle θ. We have

$$\mathtt{R}^j_{\mathrm{C}} = \mathcal{R}(\mathbf{a}, \frac{2\pi}{n} j)\mathtt{R}^0_{\mathrm{C}} \text{ if } j \in \{1, \dots, n-1\} \tag{5.3}$$

where $\mathtt{R}^0_{\mathrm{C}}$ is chosen with respect to the symmetry axis $\mathbf{a}$. This holds for all our multi-cameras (four GoPro, LadyBug, Ricoh ThetaS) for a value of n ($n \in \{4, 5, 2\}$), see Figure 5.3. In this figure, the camera pointing directions are perpendicular to $\mathbf{a}$.

Now, we have a complete initialization of the multi-camera calibration and apply a multi-camera SfM based on keyframe subsampling of the video (see Section 2.4) to obtain a 3D geometry initialization. The parameters including both intrinsic and extrinsic parameters, 3D geometry (key-frame poses and 3D structure) are refined by a global multi-camera BA in the next sections. We remind that the *keyframes* are the only

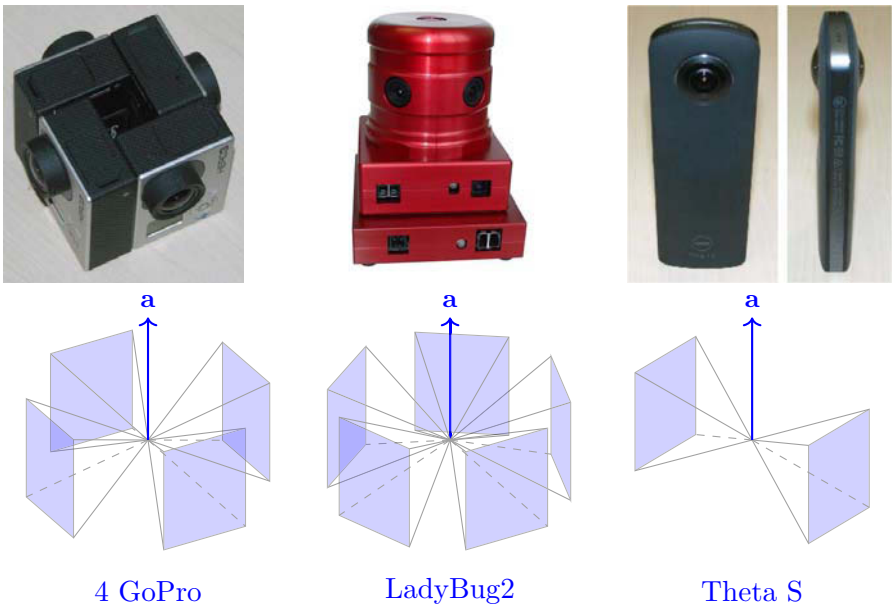


FIGURE 5.3: Our multi-cameras and initialization of extrinsic parameters. A pyramid represents camera's FoV. Cameras are symmetrically mounted around a symmetry axis **a**. For readability of the figure, the FoVs have empty intersections.

frames whose poses are refined by BA. This is useful for both time computation and accuracy.

5.3 Reprojection error and its derivatives

Subsection 5.3.1 computes the reprojection error minimized by BA. Subsection 5.3.2 introduces some notations for derivatives of reprojection error with respect to a vector of parameters optimized by BA.

5.3.1 Cost function

Let $\mathbf{I}^j$ be the vector of intrinsic parameters of the j-th camera, $\mathbf{X}^l = (X\ Y\ Z\ w)^\top$ be a 3D point of the scene $\mathcal{X} = \{\mathbf{X}^l\}$ expressed in the world coordinate system. First, by passing from the world coordinate system to the multi-camera coordinate system with the transformation $\mathtt{T}^i_{\mathrm{M}}(\mathtt{R}^i_{\mathrm{M}}, \mathbf{t}^i_{\mathrm{M}})$, and then passing from the multi-camera coordinate system to the camera coordinate system with the transformation $\mathtt{T}^j_{\mathrm{C}}(\mathtt{R}^j_{\mathrm{C}}, \mathbf{t}^j_{\mathrm{C}})$, the inhomogeneous coordinate $\tilde{\mathbf{X}}_{\mathrm{C}}$ of this 3D point in the j-th camera coordinate system meets

$$\tilde{\mathbf{X}}_{\mathrm{C}} = (\mathtt{R}^j_{\mathrm{C}})^\top \left[(\mathtt{R}^i_{\mathrm{M}})^\top \mid \quad (\mathtt{R}^i_{\mathrm{M}})^\top \mathbf{t}^i_{\mathrm{M}} \quad \mathbf{t}^j_{\mathrm{C}} \right] \mathbf{X}^l. \tag{5.4}$$

The forward projection from $\tilde{\mathbf{X}}_{\mathrm{C}}$ to 2D image point $\mathbf{p}$ is detailed in Section 4.1. So the projected point $\mathbf{p}$ is a function of the vector of parameters $\boldsymbol{\theta} = \{\mathbf{X}^l, \mathtt{R}^i_{\mathrm{M}}, \mathbf{t}^i_{\mathrm{M}}, \mathtt{R}^j_{\mathrm{C}}, \mathbf{t}^j_{\mathrm{C}}, \mathbf{I}^j\}$. Let $\tilde{\mathbf{p}}$ be the observation of the 3D point $\mathbf{X}^l$ in the j-th sub-image (provided by the j-th camera) of the i-th keyframe. We define

$$\boldsymbol{\epsilon}^l_{ij} = \mathbf{p}(\boldsymbol{\theta}) \quad \tilde{\mathbf{p}} \tag{5.5}$$

as the reprojection error of 3D point $\mathbf{X}^l$ observed by the j-th camera at the i-th keyframe pose. The cost function is

$$F = \frac{1}{2}\sum_{(i,j,l)} (\boldsymbol{\epsilon}_{ij}^l)^\top \boldsymbol{\epsilon}_{ij}^l = \frac{1}{2}\boldsymbol{\epsilon}^\top \boldsymbol{\epsilon} \tag{5.6}$$

where $\boldsymbol{\epsilon}$ is the residual vector concatenating all vectors $\boldsymbol{\epsilon}_{ij}^l$. Under the standard assumption that the image noise due to point detection follows zero-mean normalized identical and independent Gaussian vectors, the result returned by BA is a Maximum Likelihood Estimation. We use the LM algorithm in order to minimize the cost function F as Eq.(5.6).

Here, we remind the computation of the projection $\mathbf{p}$ for the polynomial distortion model that does not have a closed-form (as discussed in Subsection 4.1.1). This model is used in [Lébraly+11] but the reprojection error is minimized in the *rectified space* (explained in the next paragraph). For the unified camera model, it is straightforward since the forward projection of this model is closed-form (This is detailed in Appendix B).

Polynomial distortion model. The projection $\mathbf{p} = p_j(\tilde{\mathbf{X}}_\mathrm{C})$ is described in Section 4.1.1.2 assuming that $\tilde{\mathbf{X}}_\mathrm{C}$ is known. Generally, $\tilde{\mathbf{X}}_\mathrm{C}$ is a function of the extrinsic parameters $(\mathtt{R}_\mathrm{C}^j, \mathbf{t}_\mathrm{C}^j)$, the multi-camera pose $(\mathtt{R}_\mathrm{M}^i, \mathbf{t}_\mathrm{M}^i)$ and the 3D point $\mathbf{X}^l$ as Eq.(5.4). The vector $\boldsymbol{\theta}$ in this chapter is a generalization of that one in Eq.(4.6). [Lébraly+11] defines the reprojection error in the rectified space as follows

$$\boldsymbol{\epsilon} = g(\tilde{\mathbf{p}}, \boldsymbol{\theta}) \tag{5.7}$$

where g is defined in Eq.(4.8). We assume that the measurement noise is zero-mean normalized identical and independent Gaussian vector and its effect on the error terms is also an approximation of a Gaussian distribution. We can see that the measurement noise in point $\tilde{\mathbf{p}}$ detection is not directly fed through to the error term in Eq.(5.7) (one can see it more clearly in Eq.(4.8) with $\mathbf{z} = \tilde{\mathbf{p}}$). In essence, the uncertainty from the image plane, i.e. the original space, needs to transfer to the rectified space. However, [Lébraly+11] assumes the covariance of the error terms is equal to identity although there are large distortions between rectified and distorted images (see Figure 5.4). Our BA minimizes the reprojection error in the original space as defined in Eq.(5.5). It allows us to use the important assumption in which the covariance matrix for $\tilde{\mathbf{p}}$ is identity. Our BA is a Maximum Likelihood Estimator (this assumption is not true in the undistorted space).

5.3.2 Analytical derivatives

In this subsection, we summarize the analytical Jacobian of the residuals with respect to the optimized parameters $\boldsymbol{\theta} = \{\mathbf{X}^l, \mathtt{R}_\mathrm{M}^i(\mathbf{r}_\mathrm{M}^i), \mathbf{t}_\mathrm{M}^i, \mathtt{R}_\mathrm{C}^j(\mathbf{r}_\mathrm{C}^j), \mathbf{t}_\mathrm{C}^j, \mathbf{I}^j\}$ where $\mathbf{r}_\mathrm{M}^i$ and $\mathbf{r}_\mathrm{C}^j$ are rotation parameters (local Euler angles or quaternion). In this chapter, we choose the usual local Euler angles [Triggs+00]. For each iteration of LM algorithm, a local rotation $\mathcal{R}(\mathbf{r})$ is composed with a current rotation $\mathtt{R}_0$ to get update rotation $\mathtt{R}(\mathbf{r}) = \mathcal{R}(\mathbf{r})\mathtt{R}_0$ where $\mathbf{r} = (\alpha\ \beta\ \gamma)^\top$ concatenates the small local Euler angles (this avoids singularities). Let

- $\mathtt{A}$ be the Jacobian matrix with respect to all extrinsic $\mathtt{T}_\mathrm{C}^j(\mathbf{r}_\mathrm{C}^j, \mathbf{t}_\mathrm{C}^j)$ and all intrinsic parameters $\mathbf{I}^j = (f_x, f_y, u_0, v_0, k_1, \ldots, k_5)$ for the polynomial distortion model or $\mathbf{I}^j = (f_x, f_y, u_0, v_0, \xi)$ for the unified camera model,

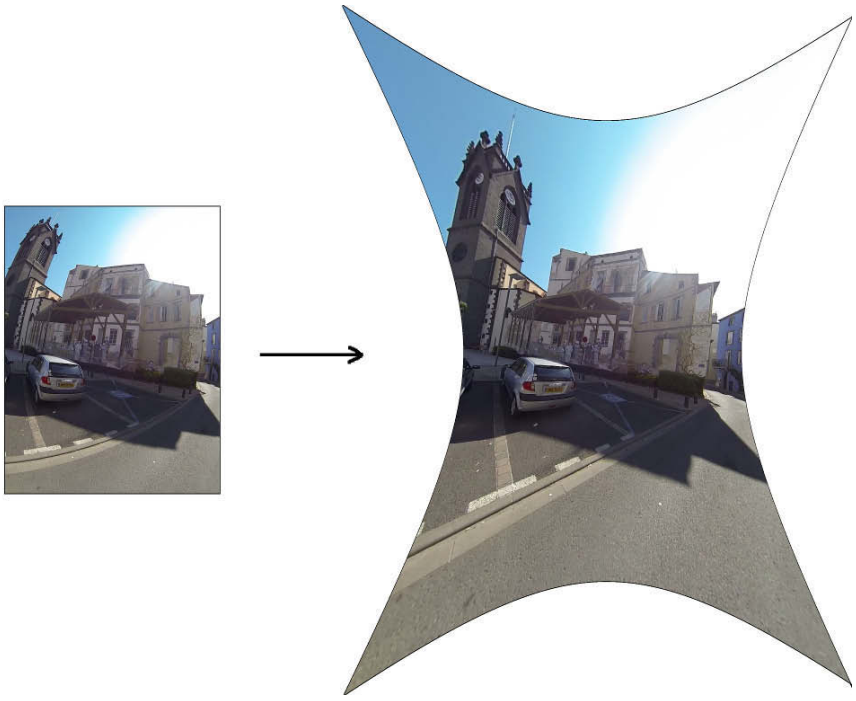


FIGURE 5.4: Example for original image (left) and rectified image (right). Original image is extracted from the video BC1 using GoPro camera.

- $\mathtt{B}$ be the Jacobian matrix with respect to all multi-camera keyframe poses $\mathtt{T}_{\mathrm{M}}^{i}(\mathbf{r}_{\mathrm{M}}^{i}, \mathbf{t}_{\mathrm{M}}^{i})$,
- $\mathtt{C}$ be the Jacobian matrix with respect to all 3D points $\mathbf{X}^{l}$.

Each elementary block of the Jacobian matrix is analytically expressed as follows:

$$\mathtt{A}_{ij}^{l} = \begin{pmatrix} \frac{\partial \boldsymbol{\epsilon}_{ij}^{l}}{\partial \mathtt{T}_{\mathrm{C}}^{j}} & \frac{\partial \boldsymbol{\epsilon}_{ij}^{l}}{\partial \mathbf{I}^{j}} \end{pmatrix} = \begin{pmatrix} \frac{\partial \mathbf{p}}{\partial \mathtt{T}_{\mathrm{C}}^{j}} & \frac{\partial \mathbf{p}}{\partial \mathbf{I}^{j}} \end{pmatrix} = \begin{pmatrix} \frac{\partial \mathbf{p}}{\partial \mathbf{r}_{\mathrm{C}}^{j}} & \frac{\partial \mathbf{p}}{\partial \mathbf{t}_{\mathrm{C}}^{j}} & \frac{\partial \mathbf{p}}{\partial \mathbf{I}^{j}} \end{pmatrix} \tag{5.8}$$

$$\mathtt{B}_{ij}^{l} = \frac{\partial \boldsymbol{\epsilon}_{ij}^{l}}{\partial \mathtt{T}_{\mathrm{M}}^{i}} = \frac{\partial \mathbf{p}}{\partial \mathtt{T}_{\mathrm{M}}^{i}} = \begin{pmatrix} \frac{\partial \mathbf{p}}{\partial \mathbf{r}_{\mathrm{M}}^{i}} & \frac{\partial \mathbf{p}}{\partial \mathbf{t}_{\mathrm{M}}^{i}} \end{pmatrix} \tag{5.9}$$

$$\mathtt{C}_{ij}^{l} = \frac{\partial \boldsymbol{\epsilon}_{ij}^{l}}{\partial \mathbf{X}^{l}} = \frac{\partial \mathbf{p}}{\partial \mathbf{X}^{l}}. \tag{5.10}$$

$\mathtt{A}_{ij}^{l}$ is a 2×15 matrix for the polynomial distortion model (6 extrinsic and 9 intrinsic parameters) or a 2×11 matrix for the unified camera model (6 extrinsic and 5 intrinsic parameters). $\mathtt{B}_{ij}^{l}$ and $\mathtt{C}_{ij}^{l}$ are respectively 2×6 and 2×3 matrices. The calculation of Eq.(5.8), Eq.(5.9) and Eq.(5.10) depends on the camera model (detail in Appendix B).

5.4 LM algorithm and normal equation

Suppose that our MCBA problem consists of n cameras, $m+1$ keyframes and p 3D points. Let Θ be the vector that concatenates all

- extrinsic parameters $(\mathtt{R}_{\mathrm{C}}^{j}, \mathbf{t}_{\mathrm{C}}^{j})$ of the j-th camera in the multi-camera coordinate system. Note that $\mathbf{t}_{\mathrm{C}}^{j}$ is optional (If the multi-camera is central, $\mathbf{t}_{\mathrm{C}}^{j} = \mathbf{0}$).

- intrinsic parameters for each camera
- multi-camera poses $(\mathtt{R}_{\mathrm{M}}^{i}, \mathbf{t}_{\mathrm{M}}^{i})$
- 3D points $\mathbf{X}^{l}$.

For every cameras, we assume that the intrinsic and extrinsic parameters are constant (i.e. the same for all keyframes). In our implementation, we use a minimal parametrization by fixing 10 (central case) or 13 (non-central case) independent parameters as detailed in Appendix C. As in the monocular case, we remove from Θ the first pose $(\mathtt{R}_{\mathrm{M}}^{0}, \mathbf{t}_{\mathrm{M}}^{0})$ and one of the 3 coordinates of one $\mathbf{t}_{\mathrm{M}}^{i}$ where $i \neq 0$ and which is not equal to that of $\mathbf{t}_{\mathrm{M}}^{0}$. Then we remove from Θ the rotation of extrinsic parameter of the first camera in the multi-camera coordinate $\mathtt{R}_{\mathrm{C}}^{0}$. In the non-central case, we also remove $\mathbf{t}_{\mathrm{C}}^{0}$.

We remind that the LM algorithm is a non-linear iterative optimization that solves the least squares problem by combining the advantages of gradient and Gauss-Newton methods. Using first order approximation, we obtain the normal equation as Eq.(2.27). The objective is to calculate the Δ that updates Θ; Δ meets

$$\mathtt{J}^{\top}\mathtt{J}\Delta = -\mathtt{J}^{\top}\boldsymbol{\epsilon} \quad \Leftrightarrow \quad \mathtt{H}\Delta = \mathbf{g} \tag{5.11}$$

where $\mathtt{J}$ is Jacobian matrix of residual $\boldsymbol{\epsilon}$ with respect to Θ, $\mathtt{H} = \mathtt{J}^{\top}\mathtt{J}$ is an approximation of the matrix Hessian and $\mathbf{g} = -\mathtt{J}^{\top}\boldsymbol{\epsilon}$. More precisely, the diagonal terms of matrix Hessian are multiplied by $(1 + 1/\mu)$ as Eq.(2.28) where $1/\mu$ is damping parameter (here we do not write this to simplify notation).

Using the notations $\mathtt{A}, \mathtt{B}, \mathtt{C}$ in Section 5.3, the Jacobian matrix is partitioned as $\mathtt{J} = [\mathtt{A}|\mathtt{B}|\mathtt{C}]$. Let $\mathtt{Q} = \mathtt{A}^{\top}\mathtt{A}$, $\mathtt{U} = \mathtt{B}^{\top}\mathtt{B}$, $\mathtt{V} = \mathtt{C}^{\top}\mathtt{C}$, $\mathtt{E} = \mathtt{A}^{\top}\mathtt{B}$, $\mathtt{F} = \mathtt{A}^{\top}\mathtt{C}$, $\mathtt{W} = \mathtt{B}^{\top}\mathtt{C}$, $\mathbf{g}_{\mathtt{A}} = -\mathtt{A}^{\top}\boldsymbol{\epsilon}$, $\mathbf{g}_{\mathtt{B}} = -\mathtt{B}^{\top}\boldsymbol{\epsilon}$ and $\mathbf{g}_{\mathtt{C}} = -\mathtt{C}^{\top}\boldsymbol{\epsilon}$. The normal equation Eq.(5.11) is expressed with block matrices

$$\left[\begin{array}{cc|c} \mathtt{Q} & \mathtt{E} & \mathtt{F} \\ \mathtt{E}^{\top} & \mathtt{U} & \mathtt{W} \\ \hline \mathtt{F}^{\top} & \mathtt{W}^{\top} & \mathtt{V} \end{array}\right] \left(\begin{array}{c} \Delta_{\mathtt{A}} \\ \Delta_{\mathtt{B}} \\ \hline \Delta_{\mathtt{C}} \end{array}\right) = \left(\begin{array}{c} \mathbf{g}_{\mathtt{A}} \\ \mathbf{g}_{\mathtt{B}} \\ \hline \mathbf{g}_{\mathtt{C}} \end{array}\right) \tag{5.12}$$

$$\Leftrightarrow \left[\begin{array}{c|c} \bar{\mathtt{U}} & \bar{\mathtt{W}} \\ \hline \bar{\mathtt{W}}^{\top} & \mathtt{V} \end{array}\right] \left(\begin{array}{c} \Delta_{\mathtt{AB}} \\ \hline \Delta_{\mathtt{C}} \end{array}\right) = \left(\begin{array}{c} \mathbf{g}_{\mathtt{AB}} \\ \hline \mathbf{g}_{\mathtt{C}} \end{array}\right). \tag{5.13}$$

where $\bar{\mathtt{U}} = \begin{bmatrix} \mathtt{Q} & \mathtt{E} \\ \mathtt{E}^{\top} & \mathtt{U} \end{bmatrix}$, $\bar{\mathtt{W}} = \begin{bmatrix} \mathtt{F} \\ \mathtt{W} \end{bmatrix}$ and increment of (intrinsic and extrinsic) camera parameters and multi-camera poses $\Delta_{\mathtt{AB}} = \begin{pmatrix} \Delta_{\mathtt{A}} \\ \Delta_{\mathtt{B}} \end{pmatrix}$. The normal equation of classical BAs (monocular or multi-camera with fixed intrinsic and extrinsic parameters) have the following form

$$\left[\begin{array}{c|c} \mathtt{U} & \mathtt{W} \\ \hline \mathtt{W}^{\top} & \mathtt{V} \end{array}\right] \left(\begin{array}{c} \Delta_{\mathtt{B}} \\ \hline \Delta_{\mathtt{C}} \end{array}\right) = \left(\begin{array}{c} \mathbf{g}_{\mathtt{B}} \\ \hline \mathbf{g}_{\mathtt{C}} \end{array}\right). \tag{5.14}$$

In the case of our MCBA, we add in the normal equations the lines and columns relative to intrinsic and extrinsic parameters. Both sides of Eq.(5.13) are multiplied by $\begin{bmatrix} \mathtt{I} & -\bar{\mathtt{W}}\mathtt{V}^{-1} \\ \mathtt{0} & \mathtt{I} \end{bmatrix}$:

$$\left[\begin{array}{c|c} \bar{\mathtt{U}} - \bar{\mathtt{W}}\mathtt{V}^{-1}\bar{\mathtt{W}}^{\top} & 0 \\ \hline \bar{\mathtt{W}}^{\top} & \mathtt{V} \end{array}\right] \left(\begin{array}{c} \Delta_{\mathtt{AB}} \\ \hline \Delta_{\mathtt{C}} \end{array}\right) = \left(\begin{array}{c} \mathbf{g}_{\mathtt{AB}} - \bar{\mathtt{W}}\mathtt{V}^{-1}\mathbf{g}_{\mathtt{C}} \\ \hline \mathbf{g}_{\mathtt{C}} \end{array}\right). \tag{5.15}$$

The matrix

$$\mathtt{S} = \bar{\mathtt{U}} - \bar{\mathtt{W}}\mathtt{V}^{-1}\bar{\mathtt{W}}^{\top} \tag{5.16}$$

is the Schur complement of $\mathtt{V}$ in $\mathtt{H}$. We have

$$\mathtt{S}\Delta_{\mathtt{AB}} = \mathbf{g}_{\mathtt{AB}} - \bar{\mathtt{W}}\mathtt{V}^{-1}\mathbf{g}_{\mathtt{C}}. \tag{5.17}$$

The matrix $\mathtt{S}$ is also known as the reduced camera/pose matrix, because the parameters (incremented by $\Delta_{\mathtt{AB}}$) in Eq.(5.17) are the ones corresponding to the cameras (multi-camera pose, intrinsic and extrinsic parameters).

Now, Eq.(5.13) can be solved by first forming $\mathtt{S}$, solving for $\Delta_{\mathtt{AB}}$ and then back-substituting $\Delta_{\mathtt{AB}}$ in order to obtain the value of $\Delta_{\mathtt{C}}$. Thus, the solution of Eq.(5.13) is reduced to the inversion of the block-wise diagonal matrix $\mathtt{V}$, a matrix-matrix and matrix-vector multiplication and the solution of block sparse linear system in Eq.(5.17). For almost all problems, the number of keyframe poses is much smaller than the number of 3D points, thus solving Eq.(5.17) is significantly cheaper than directly solving Eq.(5.13). The matrix $\mathtt{S}$ is typically a fairly sparse matrix, as most keyframes only see a small fraction of the scene. Ceres solver [Agarwal+] stores $\mathtt{S}$ as a sparse matrix, uses row and column re-ordering algorithms to maximize the sparsity of the Cholesky decomposition and focus the computational effort on the non-zero part of the factorization. The LM method is summarized in Algorithm 2, Section 2.3.

5.5 Degenerate camera motions

This section summarizes known degenerate motions where the MCBA provides ambiguous results. If a degenerate motion occurs, the calibration is only partial: some extrinsic parameters can not be estimated. In other words, changing these parameters does not change the value of minimized cost function.

As the linear extrinsic calibration as Eqs.(3.23) and (3.24) is the same as hand-eye problem, we can deduce degenerate cases from the previous studies [Andreff+01], [Fassi+05]. In the context of motion and structure from stereo images without stereo correspondence, [Kim+06] reports a degenerate case such that the axes $\mathbf{n}_{\mathrm{M}}^{i}$ of the rotations of multi-camera keyframe poses are parallel. The pure translations reported in [Esquivel+07] are also degenerate cases. Using the representation of rotations by orthogonal matrices as suggested in [Andreff+01], the work in [Lébraly+10a] outlines several degenerate motions of extrinsic calibration as illustrated in Table 5.1. Since we do both intrinsic and extrinsic calibration, these degenerate motions also occur in our case. The number of observable DoF of the extrinsic parameters of the non-overlapping multi-cameras is also represented.

A practically important case is planar motion where $\mathtt{R}_{\mathrm{M}}^{i}$ have parallel axes $\mathbf{n}_{\mathrm{M}}^{i}$ which are orthogonal to the plane of motion. One dimension of the extrinsic translation $\mathbf{t}_{\mathrm{C}}^{j}$ is only estimated up to an unknown scale along the normal to the camera plane of motion. In our context, we deal with omnidirectional multi-camera. In the central case, we fix the extrinsic translation parameters $\mathbf{t}_{\mathrm{C}}^{j} = \mathbf{0}$. Under planar motion, the extrinsic rotation parameter $\mathtt{R}_{\mathrm{C}}^{j}$ can be fully estimated. In the non-central case, we can not determine the component perpendicular to the camera plane of motion. Our helmet-held multi-camera motions are close to this case, but thanks to pose noise (at least small rotation

Motions	Axes of rotation	$\mathtt{R}_{\mathrm{C}}^{j}$	$\mathbf{t}_{\mathrm{C}}^{j}$
1) Rotations and screw motions about an axis $\mathbf{a}$	same and equal to $\mathbf{a}$	2	0
2) Pure translation along an axis $\mathbf{a}$	no rotation	2	0
3) Pure translation along several axis	no rotation	3	0
4) Planar and screw motions about several axes	different and parallel	3	2
5) General case	different	3	3

TABLE 5.1: Number of observable degrees of freedom of the extrinsic parameters of multi-camera system [Lébraly+10a]. The scale is assumed known.

of the head that we use to synchronize the cameras in Section 4.2) we avoid this critical motion. [Lébraly+10a] proposes a linear solution to handle this degenerate case using constraints provided by the scene permutation. In other words, with at least one 3D point of the scene seen by each camera at different times, the component perpendicular to the camera plane of motion is calculable. As a result, the extrinsic calibration is fully estimated.

5.6 Experiments

In this section, our BA is named by a combination of several notations that describes the estimated parameters:

- C (central approximation) estimates all $\mathtt{R}_{\mathrm{C}}^{j}$ and fixes all $\mathbf{t}_{\mathrm{C}}^{j} = \mathbf{0}$,
- NC (non-central) estimates all $(\mathtt{R}_{\mathrm{C}}^{j}, \mathbf{t}_{\mathrm{C}}^{j})$,
- INT (intrinsic) estimates all intrinsic parameters: $(f_x, f_y, u_0, v_0, k_1, \ldots, k_5)$ or $(f_x, f_y, u_0, v_0, \xi)$ depending on the camera model (every camera has its own parameters),
- FA (frame accurate) is used to distinguish synchronization method in Chapter 4 from that in Chapter 6,
- GS (global shutter) is used to distinguish from rolling shutter in Chapter 6.

Thus GS.NC.FA.INT (or gs.nc.fa.int) is a BA that estimates simultaneously all extrinsic parameters $(\mathtt{R}_{\mathrm{C}}^{j}, \mathbf{t}_{\mathrm{C}}^{j})$ and intrinsic parameters and keyframe poses $(\mathtt{R}_{\mathrm{M}}^{i}, \mathbf{t}_{\mathrm{M}}^{i})$ and 3D points. The threshold for inlier selection is set to 4 pixels in all videos. A BA has three inlier updates, each one is followed by Levenberg-Marquardt minimization for these inliers. Table 3.1 summarizes our main datasets used in this chapter.

5.6.1 How to compare two calibrations?

We define the error of the estimated multi-camera calibration by a single number d which is the RMS for all multi-camera pixels of the angle between rays of the two calibrations (the estimated one and the ground truth one) that back-project the same pixel. There are two reasons to do this. First, the accuracy is only needed for the ray directions in applications (SfM, video stitching, 3D modeling of a scene) in the central

case. Second, parameters can compensate themselves if their estimations are biased (e.g. the rotation/ principal point near-ambiguity for one view [Agapito+01]).

We detail the computation of d. Since the rays of the estimated calibration and the rays of the ground truth (GT) calibration can be expressed in different coordinate systems, we estimate a registration between both coordinate systems before computing angles between rays. The registration is defined by a rotation $\mathtt{R}$ which maps one ray set to the other ray set (ignoring translation of ray origins). More precisely, $\mathtt{R}$ is minimizer of

$$e(\mathtt{R}) = \sum_{i=1}^{N} ||\mathbf{r}_i^{\text{gt}} - \mathtt{R}\mathbf{r}_i^{\text{est}}||^2 \tag{5.18}$$

where $\mathbf{r}_i^{\text{est}}$ (respectively, $\mathbf{r}_i^{\text{gt}}$) is the ray direction of the i-th pixel by the estimated (respectively, GT) multi-camera calibration. Our distance is defined by

$$d = \sqrt{\frac{e(\mathtt{R})}{N}} \tag{5.19}$$

where N is the number of (sampled) rays in a multi-camera image. Note that d is expressed in radians if $d \ll 1$; we always convert it in pixels by dividing it by the angular resolution r in Table 3.1.

5.6.2 Polynomial distortion model: original vs. rectified errors

First of all, we remind that the polynomial distortion model in Subsection 4.1.1 does not have closed-form for forward projection. The rectified errors are defined in Eq.(5.7). And the original errors are defined in Eq.(5.5) where the projected point $\mathbf{p}$ is defined by the implicit function ψ in Subsection 4.1.1.2. BA minimizes the reprojection errors for all inlier observations $\tilde{\mathbf{p}}$, i.e. $||\psi(\boldsymbol{\theta}) - \tilde{\mathbf{p}}|| \leq 4$ pixels.

Error	init	Method	#2D	RMS	d
Rectified	$72r$	gs.c.fa.int	213335	1.216	9.575
	pat	gs.c.fa	213015	1.225	1.023
Distorted	$72r$	gs.c.fa.int	213495	0.932	1.683
	pat	gs.c.fa	213108	0.946	1.023

TABLE 5.2: Comparing accuracy of gs.c.fa.X using rectified and distorted reprojection errors on 2k first frames of CC (reminder: d is in pixels). The number of keyframes is 198, #2D is number of 2D inliers.

Table 5.2 compares the calibrations obtained by minimizing the reprojection errors in the original (i.e. distorted) and rectified (i.e. undistorted) image spaces using GS.C.FA.INT applied to the 2k first frames of the CC video which has ground truth provided by manufacturer (as a table of rays). Column "init" gives details on the initialization of the calibration. "$72r$" means that we use an initial FoV angle equals to $360^o/5$ for monocular camera with refinement by a global BA (see Table 4.1), hence "r". And "*pat*" means that we use the calibration estimated using a planar calibration pattern [Lavest+98]. Although the numbers of 2D inliers are similar, the calibration error d of the distorted case is quite better (about 6 times smaller) than that of the rectified one. Such a difference can be explained as follows: there are large distortions between

rectified and distorted images (see Figure 5.4), the BAs minimize errors in different image spaces (rectified and distorted), and the distorted space is the right one to obtain a Maximum Likelihood Estimator. We also provide the numbers of 2D inliers and RMS of GS.C.FA that enforces calibration "*pat*" during BA; our RMS and inliers are slightly better but the calibration error d of "*pat*" is the best. In the remainder of this thesis, we always use the reprojection error in the original space.

5.6.3 Central vs. non-central calibrations

As a reminder, the central approximation assumes that all cameras have the same point as center, i.e. all light rays (half-lines) of the multi-camera system go across a single point. Some applications like 360 video need a central calibration. In our context, we use (and build in some cases) a central or slightly non-central omnidirectional multi-camera. So, we need to experiment with synthetic and real datasets; and show affections of the central approximation. We also remind that the BA input is obtained as follows (see Overview of our framework): initialization of multi-camera calibration on the video beginning (SfM and GS.C.FA.INT applied to the 2k first frames) followed by multi-camera SfM applied to the whole video (see Table 5.3).

Name (short name)	Camera	fr	kfr	#Tracks
BikeCity1 (BC1)	4*Gopro 3	50.4k	2047	343k
WalkTown (WT)	4*Gopro 3	70.3k	1363	240k
FlyHill (FH)	4*Gopro 3	8.6k	627	432k
BikeCity2 (BC2)	4*Gopro 3	12.5k	225	51k
CarCity (CC)	Ladybug 2	7.7k	891	282k
WalkUniv (WU)	Theta S	29.4k	1287	154k

TABLE 5.3: Datasets: numbers of frames fr, number of keyframes kfr, number of tracks #Tracks (3D points) obtained by multi-camera SfM. (see more in Table 3.1)

5.6.3.1 Ground truth - LadyBug and synthetic dataset - BC2

Table 5.4 compares calibration error d (and RMS and 2D inliers) obtained using GS.C.FA.INT, GS.NC.FA.INT, GS.C.FA and GS.NC.FA applied to videos BC2 and CC. First, GS.C.FA.INT provides a better (smaller) d than GS.C.FA since the intrinsic parameters (INT) are estimated from a longer video. The comparison between GS.NC.FA.INT and GS.NC.FA is similar. Second, we compare GS.C.FA and GS.NC.FA and see that the non-central refinement (NC) changes almost nothing if INT is not refined. If INT is refined, the NC refinement improves the BC2 calibration but it does not improve (even degrades) the CC calibration. We interpret this result as follows: the central approximation is more tenable for CC than for BC2, since CC has a larger ratio between camera-scene distance and baseline (the inter-camera distance) than BC2. We also note that the RMS and 2D inliers are similar in all cases. Figures 5.5 and 5.6 show a top view of the reconstruction of the method in Table 5.4.

Figure 5.7 depicts the centers of LadyBug2 that form a pentagon in ground truth (see Figure 5.7(b)). By construction, camera centers $\mathbf{t}_{\mathrm{C}}^{j}(t_x\ t_y\ t_z)^\top$ almost lie in a plane. So, we find a plane $\boldsymbol{\pi}$ that fits the set of the camera centers. First, we calculate the mean vector $\mathbf{t}_m$ of this set and covariance matrix $\mathtt{C}_{\mathbf{t}_{\mathrm{C}}} = \sum_j (\mathbf{t}_{\mathrm{C}}^{j} - \mathbf{t}_m)(\mathbf{t}_{\mathrm{C}}^{j} - \mathbf{t}_m)^\top$. Second,

	BC2			CC		
Method	d	RMS	#2D	d	RMS	#2D
gs.c.fa	3.379	0.728	204k	1.685	0.938	965k
gs.c.fa.int	2.018	0.723	204k	1.173	0.938	965k
gs.nc.fa	3.397	0.727	204k	1.684	0.938	965k
gs.nc.fa.int	1.417	0.723	204k	1.313	0.937	965k

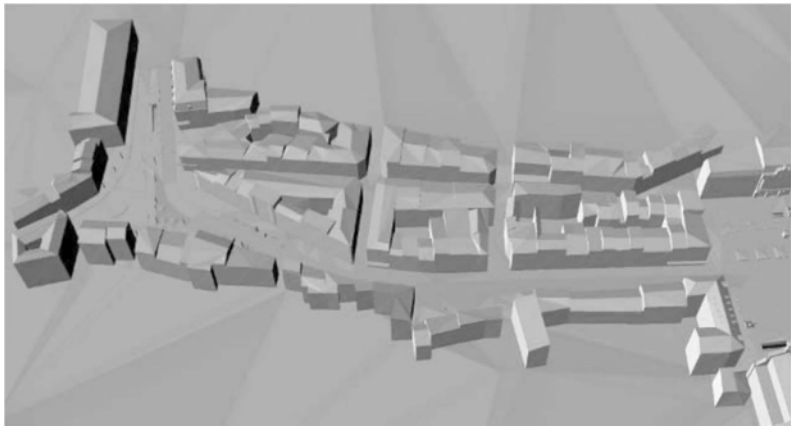
TABLE 5.4: Accuracies of gs.fa.X for BC2 (left) and CC (right).

the plane $\boldsymbol{\pi}$ is determined by $\mathbf{t}_m$ and a nonzero vector orthogonal $\mathbf{n}$ to it, i.e. the equation of $\boldsymbol{\pi}$ is $\mathbf{n}^\top(\mathbf{t}-\mathbf{t}_m)=0$. The vector $\mathbf{n}$, called the normal vector, is equal to the eigenvector corresponding to the smallest eigenvalue of the matrix $\mathtt{C}_{\mathbf{t}_\mathrm{C}}$ using Singular Value Decomposition method. The plane $\boldsymbol{\pi}$ established in this way is a best-fit plane of the set of the camera centers. We also depict the projected camera centers on the plane $\boldsymbol{\pi}$ in Figure 5.7.

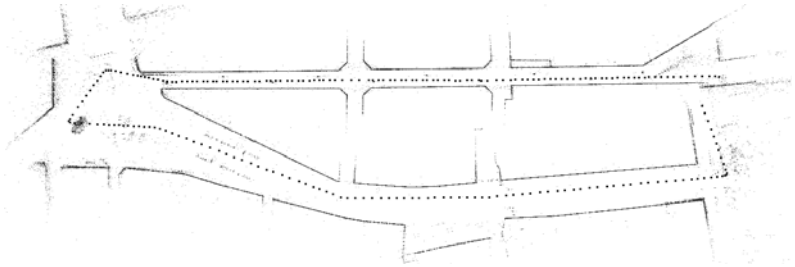
Table 5.5 provides accuracies of the intrinsic parameters of the first BC2 camera using GS.C.FA.INT and GS.NC.FA.INT. The absolute errors of f_x, f_y, u_0, v_0 are about 2 pixels or less; the relative errors of k_1 and k_2 are good and those of k_i are bad if $i > 2$. The NC-values of f_x, f_y and u_o are slightly better than those of C.

		gs.c.fa.int		gs.nc.fa.int	
	G.T.	value	error	value	error
f_x	580.773	582.809	2.037	581.296	0.523
f_y	581.266	582.844	1.578	582.605	1.339
u_0	640.827	640.989	0.162	640.978	0.151
v_0	469.056	471.305	2.249	471.540	2.284
k_1	0.368	0.368	7e-4	0.368	6e-4
k_2	0.067	0.063	0.061	0.062	0.062
k_3	0.013	0.026	1.001	0.025	0.970
k_4	0.002	-0.013	6.463	-0.013	6.378
k_5	0.013	0.018	0.434	0.018	0.420

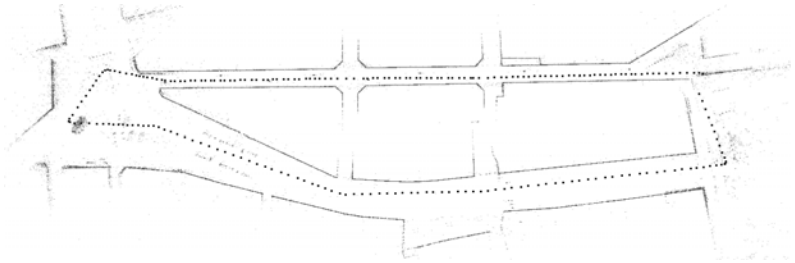
TABLE 5.5: Accuracies of intrinsic parameters of the first camera (BC2) using gs.X.fa.int. We show the absolute error in pixels for (f_x, f_y, u_0, v_0) and the relative error for k_is.



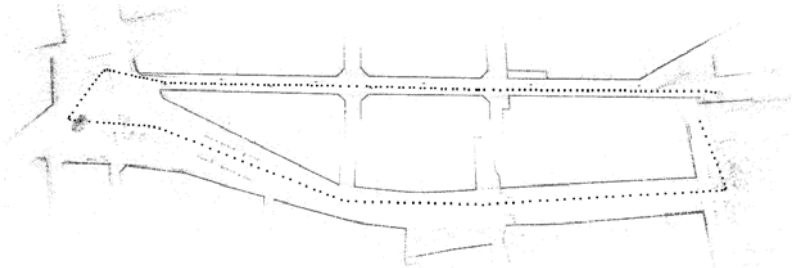
(a) Synthetic dataset



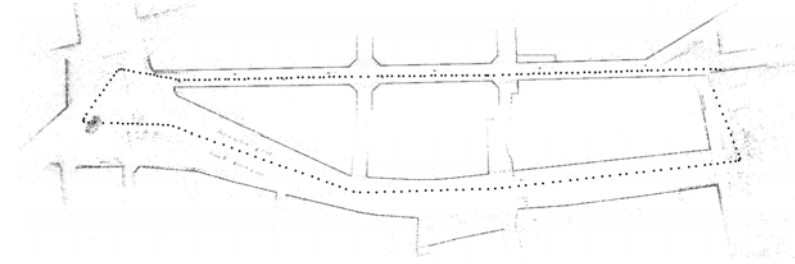
(b) gs.c.fa



(c) gs.c.fa.int

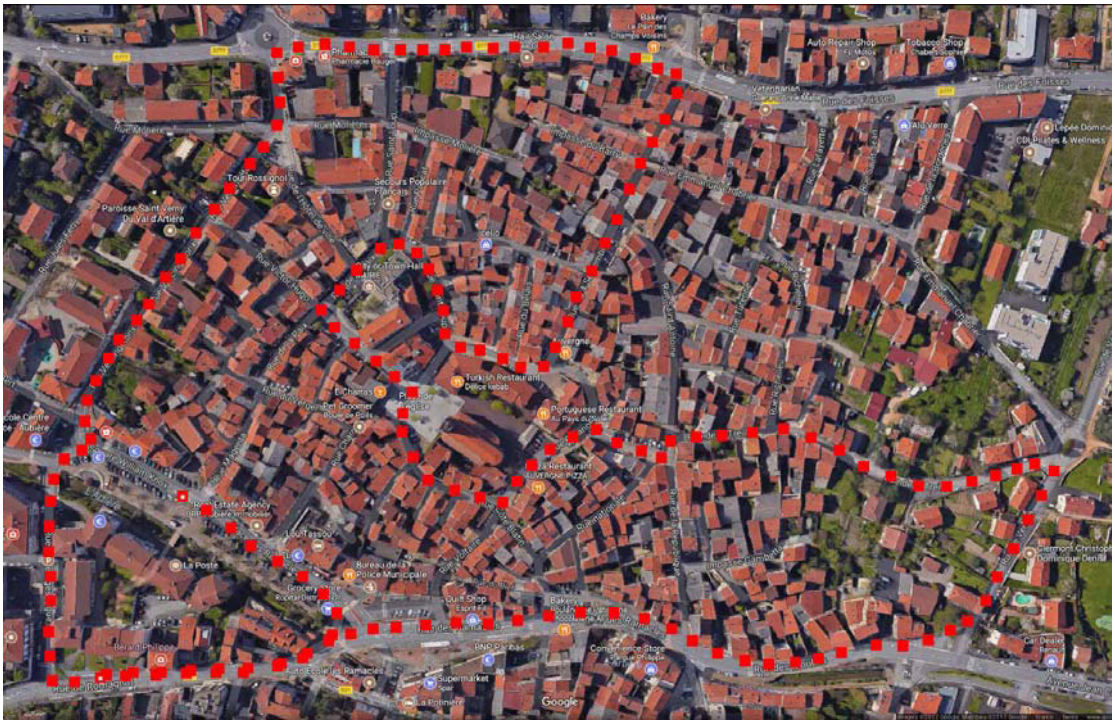


(d) gs.nc.fa

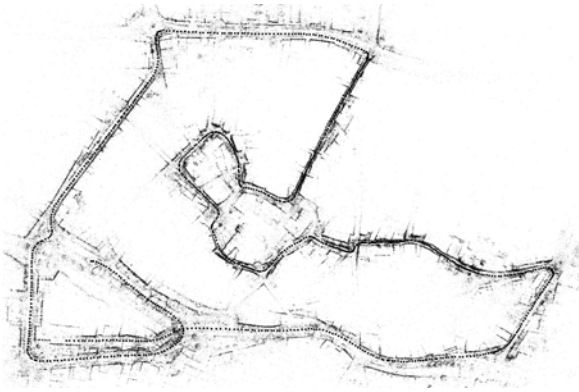


(e) gs.nc.fa.int

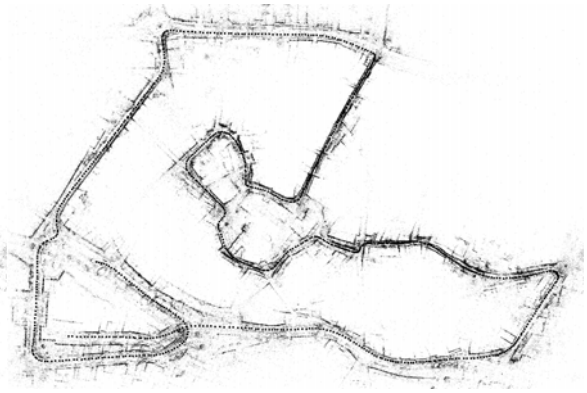
FIGURE 5.5: Top views of gs.X.fa.Y reconstruction of BC2.



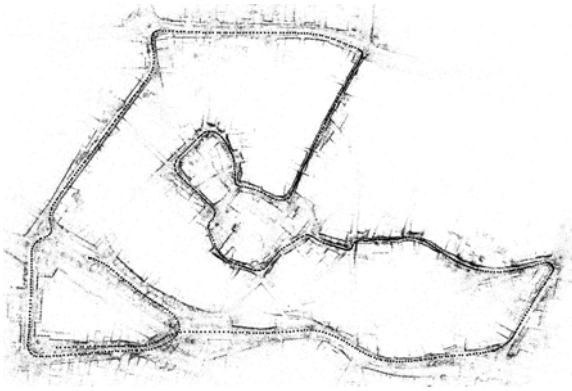
(a) Map and trajectory (best viewed in the electronic version)



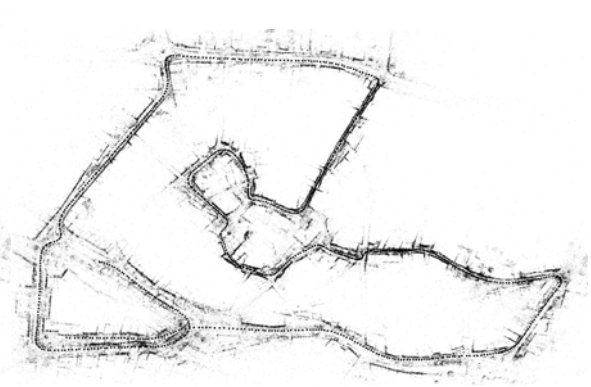
(b) gs.c.fa



(c) gs.c.fa.int

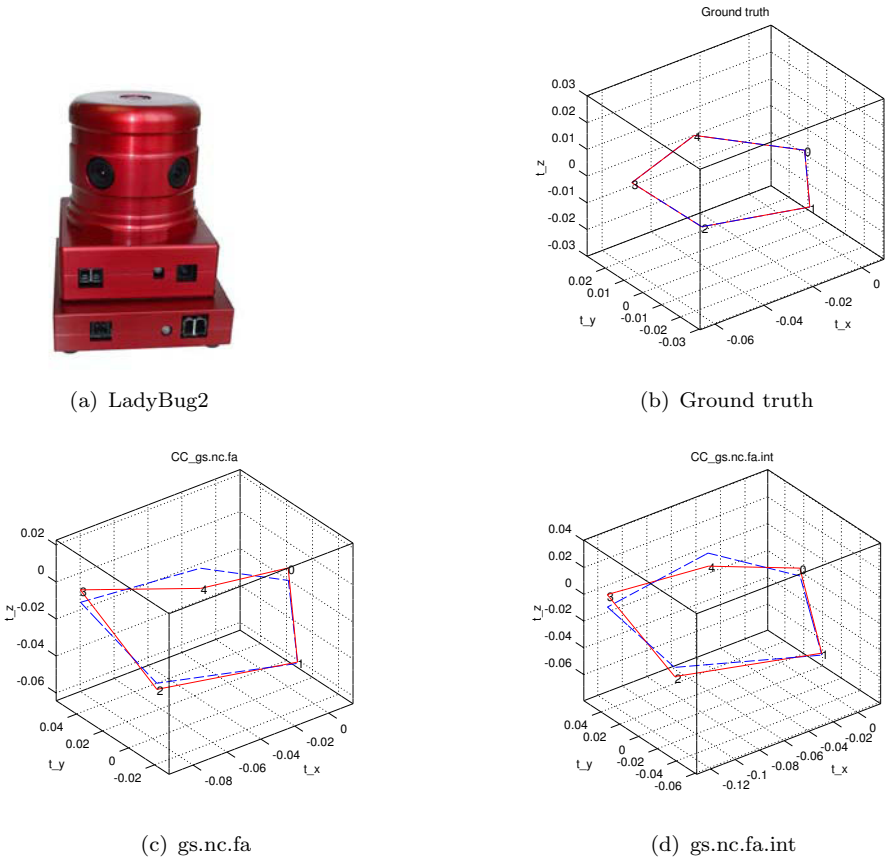


(d) gs.nc.fa



(e) gs.nc.fa.int

FIGURE 5.6: Top views of gs.X.fa.Y reconstruction of CC. The input video is taken by LadyBug2 mounted on platform of a car.



(a) LadyBug2 (b) Ground truth

(c) gs.nc.fa (d) gs.nc.fa.int

FIGURE 5.7: Non-central calibration using GS.NC.FA.INT method for CC (LadyBug2). Camera positions in multi-camera coordinate (solid red line) and its orthogonal projection on the plane $\boldsymbol{\pi}$ (dashed blue line). (see explanation in the text)

5.6.3.2 4 GoPro cameras and Theta S on a helmet

We compare the results of our method applied to three real different sequences taken by same multi-camera (4 Gopro in a cardboard): BC1 - bike riding in a city, WT - walking in a town and a short sequence CS (Cap-Sicié) - walking around an abandoned stone house. The sequence CS has the same resolution and FpS in BC1 and WT. For this sequence (with 15.5k frames), using multi-camera SfM, we obtain 784 keyframes and 140k 3D points.

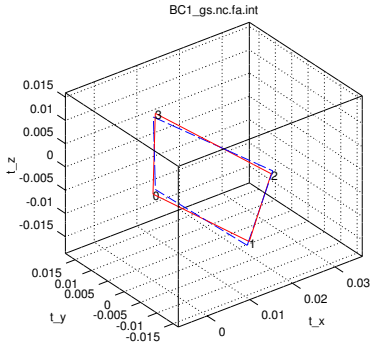
In terms of the ratio between the camera-scene distance and the inter-camera distance, BC1 is larger than WT and CS is smallest. Table 5.6 shows the initial and final bundle with or without central approximation. For points 3D and 2D inliers (#3D and #2D) and RMS, the relative error δ between two results r_{c} and r_{nc} is defined as $2(r_{\mathrm{nc}}-r_{\mathrm{c}})/(r_{\mathrm{nc}}+r_{\mathrm{c}})$. As can be seen in Table 5.6, the difference between two estimations (central and non-central calibration) is negligible for BC1. On the contrary, for CS sequence, the non-central assumption gives more significant improvement than the central assumption. The larger ratio between the scene-camera distance and the inter-camera distance, the better central approximation.

Figure 5.8 illustrates centers of Gopro multi-camera rig. In this setup, four cameras have the same configuration and their centers form a square. In the same way of CC case, we establish a plane $\boldsymbol{\pi}$ that fits the set of the camera centers. The projected

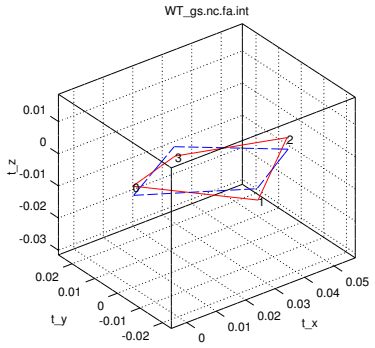
	BC1			WT			CS		
Method	#3D	#2D	RMS	#3D	#2D	RMS	#3D	#2D	RMS
initial	325k	1579k	0.830	232k	1099k	0.808	136k	619k	0.864
gs.c.fa.int	+408	+6748	0.815	+600	+3985	0.790	+236	+2002	0.851
gs.nc.fa.int	+435	+6904	0.814	+645	+4326	0.783	+351	+3188	0.818
δ (%)	0.01	0.01	-0.19	0.02	0.03	-0.85	0.08	0.19	-3.95

TABLE 5.6: Results of gs.c/nc.fa.int for BC1, WT and CS. #3D is the number of points 3D inliers, #2D is the number of points 2D inliers and δ is the relative error between two results. Note that δ is multiplied by 100 to obtain a percentage.

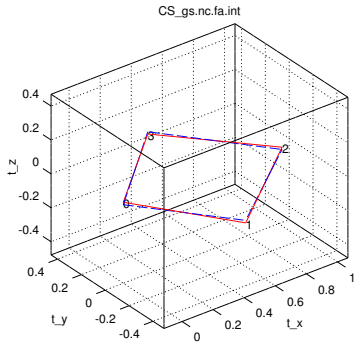
(a) 4 Gopro



(b) BC1



(c) WT

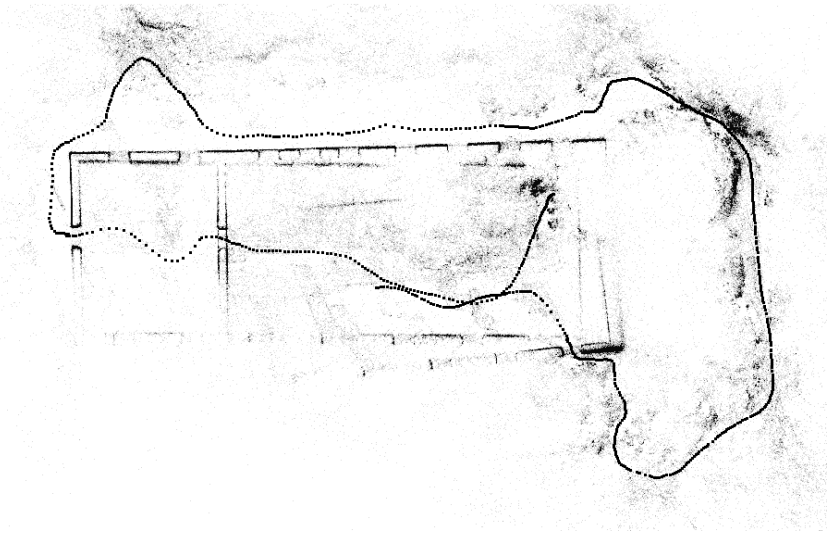


(d) CS

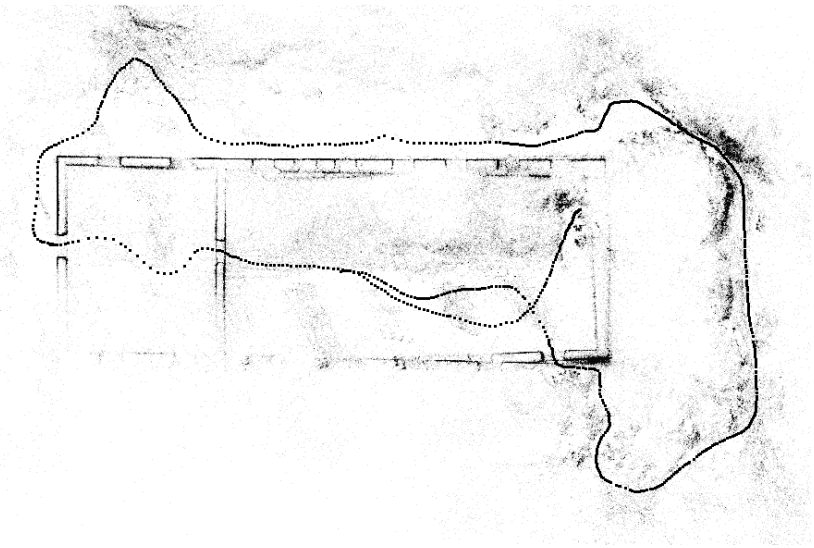
FIGURE 5.8: Non-central calibration using GS.NC.FA.INT method for BC1, WT and CS (four GoPro). Camera positions in multi-camera coordinate (solid red line) and its orthogonal projection on the plane $\boldsymbol{\pi}$ (dashed blue line). (see explanation in the text)

camera centers onto this plane $\boldsymbol{\pi}$ should be in a square form. Moreover, as mentioned in Section 5.4, we fix the extrinsic parameters $(\mathtt{R}^0_{\mathrm{C}}, \mathbf{t}^0_{\mathrm{C}})$ of the first camera. In our MCBA, we choose pointing direction of the first camera being perpendicular to the YZ plane in Figure 5.8. By construction in Figure 5.8(a), acute angle between vector $\mathbf{t}^1_{\mathrm{C}} - \mathbf{t}^0_{\mathrm{C}}$ and YZ plane should be equal to $\pi/4$. From these geometric intuitions, the non-central calibration estimation in the case of CS is noticeably close to the expected form. Furthermore, Figure 5.9 shows a top views of CS reconstruction without loop closure. We can see the improvement by the non-central calibration in the case of CS. The drift in rotation and translation is reduced, especially the wall at the bottom of this figure.

Last, the top views of the GS.C.FA.INT resulting keyframes locations and clouds



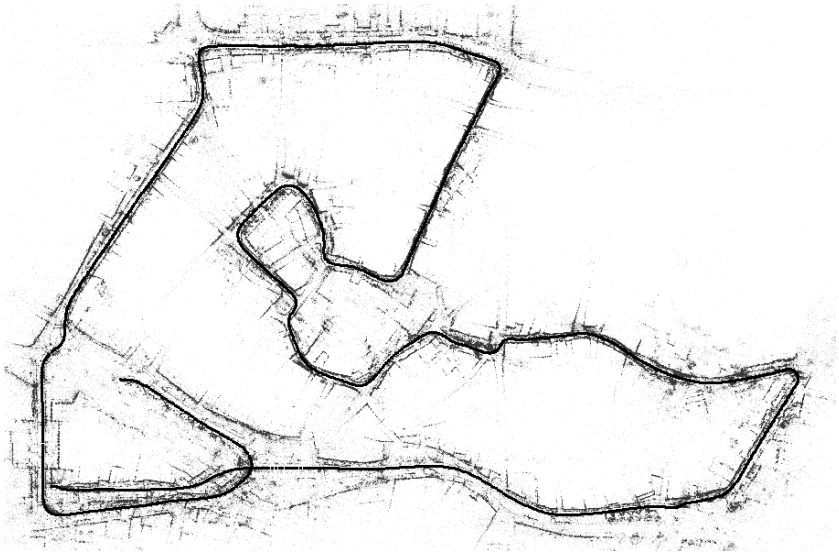
(a) Central



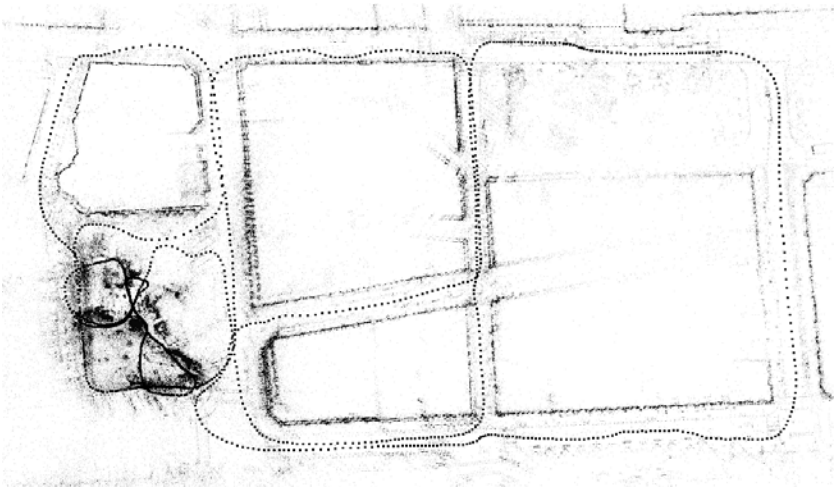
(b) Non-central

FIGURE 5.9: Top views of GS.X.FA.INT reconstruction of CS without loop closure. The input video of CS are taken by four GoPro cameras mounted on a helmet.

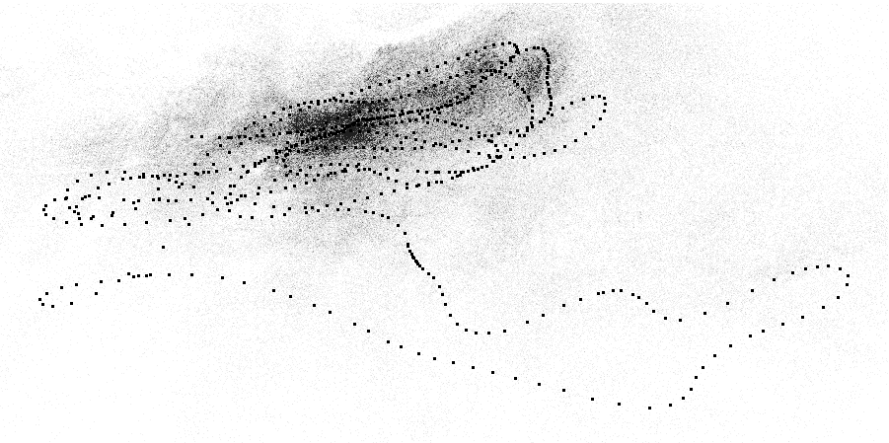
of points can be seen on Figures 5.10 and 5.11 for BC1, WT, FH (four GoPro cameras) and WU (Theta S). In the WU case, the beginning and end of the trajectory should be the same, but we observe an important drift in Figure 5.11(a). The drift is less noticeable in the other examples. Here, we do not enforce loop closure. We remind that the incremental multi-camera SfM by local BA [Mouragnon+09] is done using an intermediate calibration computed from only 2k first frames, and we see that the final BA (GS.C.FA.INT) does not remove the drift. Then, we redo the incremental SfM using the final multi-camera calibration (computed from the whole sequence by GS.C.FA.INT) and see that the important part of drift is removed. This confirms that the GS.C.FA.INT calibration refined on the whole video is better than the intermediate one.



(a) BC1

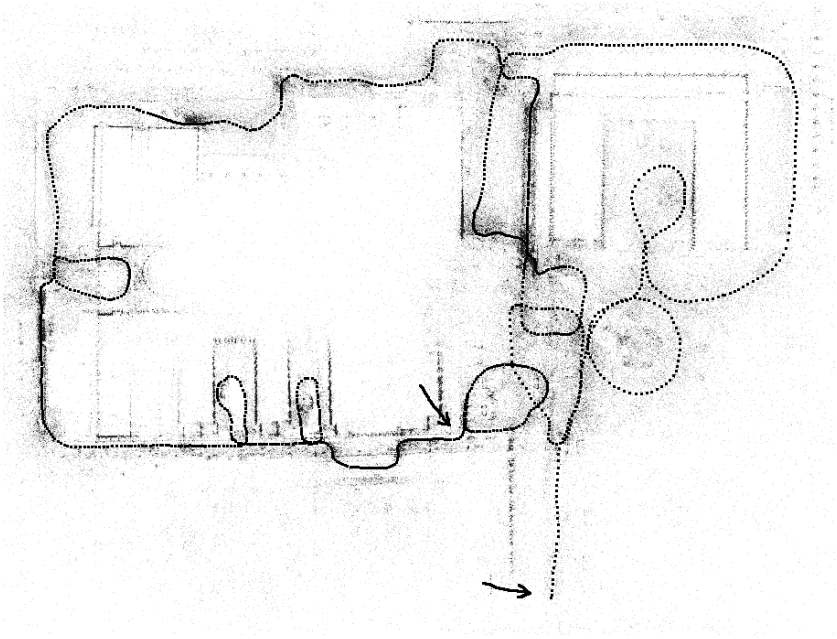


(b) WT

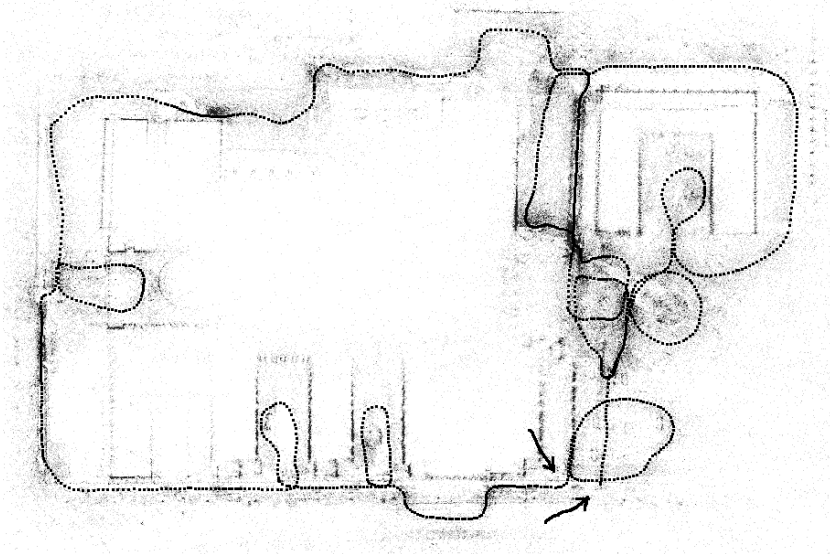


(c) FH

FIGURE 5.10: Top views of GS.C.FA.INT reconstruction of BC1, WT and FH without loop closure. The input videos of BC1, WT and FH are taken by four GoPro cameras mounted on a helmet. The FH trajectory has a lot of sharp "S" turns.



(a) GS.C.FA.INT



(b) Incremental SfM [Mouragnon+09] using the calibration estimated by GS.C.FA.INT

FIGURE 5.11: Top views of reconstruction of WU without loop closure. The input video is taken by the Ricoh Theta S mounted on a helmet. The drift is between the two arrows.

5.6.4 Global shutter approximation and varying camera speed for RS camera

In this chapter, we do assumptions: global shutter and frame-accurate synchronization. However, the cameras in our experiments are in fact rolling shutter (see Section 3.4), except for LadyBug2. The GS approximation is acceptable if the camera motion is not too fast and the shutter rolls fast enough. In other words, the image deformations due to rolling shutter should be moderated. Here, we check whether our

calibration method with GS assumption still works in the case of faster motion than before. We experiment on synthetic datasets.

In the whole of our experiments, we use synthetic sequence BC2 (about 18km/h) which mimics bicycle motion in BC1 without motion blur. In this subsection, we generate new synthetic sequences BC2 such that the mean of the speed is about 2-5 times larger than 18km/h (we also slow down in shape turns). Table 5.7 shows the results of our methods applied to these examples in terms of the calibration error d. We note that the error d increases with respect to speed by using GS.X.FA.INT which is consistent with the fact that the global shutter approximation is less tenable for high speeds. We also observe that GS.NC.FA.INT results are worse than those of GS.C.FA.INT when camera speed is greater than 40km/h. Besides rolling shutter effect, there is also subframe synchronization (see Figure 4.7) between cameras. These effects provide time-varying relative poses between sub-images in each keyframe. These results also imply that our standard SfM still works with these fast motions although it assumes that the cameras are GS.

Speed	kfr	Method	d
18km/h	225	gs.c.fa.int	2.018
		gs.nc.fa.int	1.417
20km/h	232	gs.c.fa.int	2.437
		gs.nc.fa.int	1.460
40km/h	211	gs.c.fa.int	3.301
		gs.nc.fa.int	3.708
60km/h	202	gs.c.fa.int	4.789
		gs.nc.fa.int	6.305
80km/h	198	gs.c.fa.int	6.519
		gs.nc.fa.int	7.149
100km/h	182	gs.c.fa.int	9.426
		gs.nc.fa.int	10.189

TABLE 5.7: Comparing accuracies of GS.X.FA.INT for BC2 with respect to multi-camera's speed: keyframes kfr and keyframe sampling $N_2 = 900$, $N_3 = 450$ (see Section 2.4.1). New BC2s are for 20-100km/h. The original BC2 is for 18km/h.

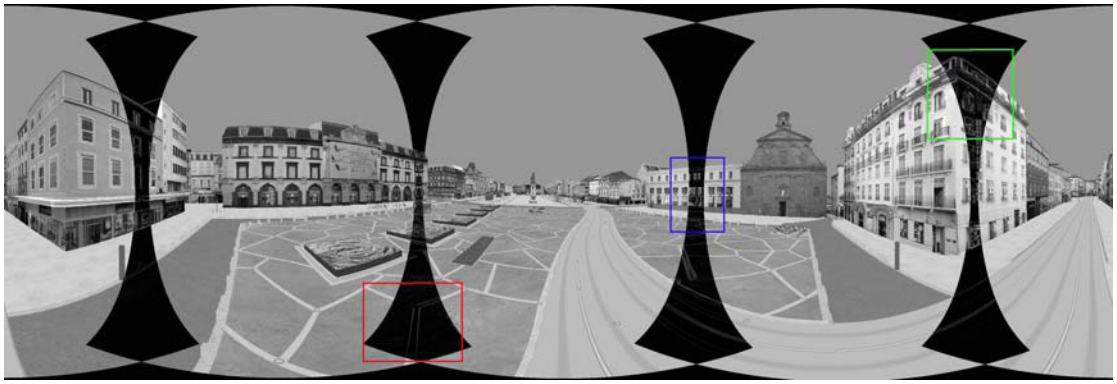
5.6.5 360 videos and 3D models

One of the applications of an omnidirectional camera is 360 video, also known as immersive video or spherical video. The video is typically recorded using a multi-camera rig where several views in several directions are filmed at the same time. Through a video stitching method, this separate footage is merged together into one cylinder video. 360 video puts users at the centre of the action and they have control of the viewing direction like a panorama during playback.

A discrepancy of a pixel in an image of a 360 video is the absolute value of the differences between two luminances captured from different cameras. Figure 5.12 shows an example of an equirectangular image without and with its discrepancies (Figure 5.12 also shows that the sharing FoV between adjacent cameras is small). The lower the discrepancy, the darker the gray level.

(a) Original images

(b) Panorama image



(c) With discrepancies

(d) Local view

FIGURE 5.12: Equirectangular image without and with discrepancies in area shared by two original images of BC2 using calibration estimated by GS.C.FA.INT. The lower the discrepancy, the darker the gray level.

Another application of our self-calibration method (assuming global shutter and frame-accurate synchronization) is 3D scene modeling from videos taken by our DIY helmet-held multi-camera. There are three examples available:

- BC1 (BikeCity1): https://skfb.ly/6tKA8
- Forest: https://skfb.ly/69p8p
- WT (WalkTown): https://youtu.be/5r46SEBvz5w

The two formers are recent examples stored in Sketchfab; the user can move interactively inside the 3D models (use the first person mode for best viewing, not the orbit mode). The latter is a companion video of [Lhuillier+15] stored in Youtube; it successively shows input images, SfM and self-calibration result (as a point cloud), and walkthroughs in the textured 3D model.

Although the 3D model computation is outside the scope of this thesis, we shortly summarize the process. In contrast to the PhD manuscript, the loops are detected and closed after the SfM step and the number of keyframes is higher. These changes improve the modeling quality for several reasons: loop closing reduces drift and avoid duplicates/deletions of segments of the reconstructed scene, and the number of reconstructed points increases when the number of keyframes increases. The self-calibration is done thanks to GS.C.FA.INT (for BC1 and WT) or GS.NC.FA.INT (for Forest) and by using the polynomial distortion model [Lavest+98], [Sturm+11]. Once a whole sequence is reconstructed and calibrated, we complete the sparse cloud of points for surface reconstruction by detecting, matching and reconstructing sampled points in the gradient edges of the keyframes. Last we apply a surface reconstruction by combining two methods [Lhuillier+13], [Litvinov+14]. More details on the computations after our self-calibration and comments on the resulting surface are given in [Lhuillier+15, Section 5.4].

5.7 Conclusion

This chapter focuses on the self-calibration of a multi-camera system. Under approximations, global shutter and frame-accurate synchronization, we first initialize the extrinsic parameters using the central approximation for translation and approximately known inter-camera rotations. Then, we extend a previous BA designed for multi-cameras using a standard polynomial distortion model: both intrinsic and extrinsic parameters are refined; the reprojection errors are minimized in the right space/original images, instead of rectified images, although the forward projection is not closed form for the polynomial distortion model. In the experiments, our method is validated with both synthetic and real data. We also check the central approximation in the real datasets and the global shutter approximation with respect to motion of RS camera. In the next chapter, we will investigate on rolling shutter and sub-frame synchronization.

Chapter 6

Multi-camera bundle adjustment adding rolling shutter and synchronization

Multi-camera rig is built by fixing together several consumer cameras. Such a multi-camera has drawbacks. A first problem is the lack of accurate synchronization between the cameras. Secondly, the low price of a consumer camera implies that the camera is rolling shutter or RS. Both inaccurate synchronization and RS complicate the self-calibration for the same reason: they act as time varying relative pose between the cameras, i.e. the multi-camera has a varying non-central calibration. In the previous chapter, we did approximations: global shutter (GS) and frame-accurate (FA) synchronization. By contrast, this chapter introduces a self-calibration method for a multi-camera moving in a static scene, that simultaneously estimates intrinsic parameters, inter-camera poses, time offsets and line delay in addition to the usual parameters (3D points and multi-camera poses). The results of this chapter are partly published in [Nguyen+16a], [Nguyen+17].

Contents

6.1 Problem formulation

Instead of a simultaneous acquisition of all pixels in an image, RS camera acquires images by scanning the image. If the camera moves in the scene, different pixels in the image have different projection centers (Figure 6.1) and this will result in geometric distortions. In many cases, for example Gopro multi-camera in our experiments, the monocular cameras record videos separately. Even though the videos are frame-accurately synchronized, there are subframe residual time offsets between the videos (Figure 6.2). The RS effect and inaccurate synchronization can degrade the accuracy of self-calibration as well as the quality of results in applications such as 3D reconstruction and 360 video. If the pose of the multi-camera is evaluated at any instant somehow, errors between prediction and observation can be calculated. Thus we require a generalized camera model for these moving RS multi-camera rigs with inaccurate synchronization.

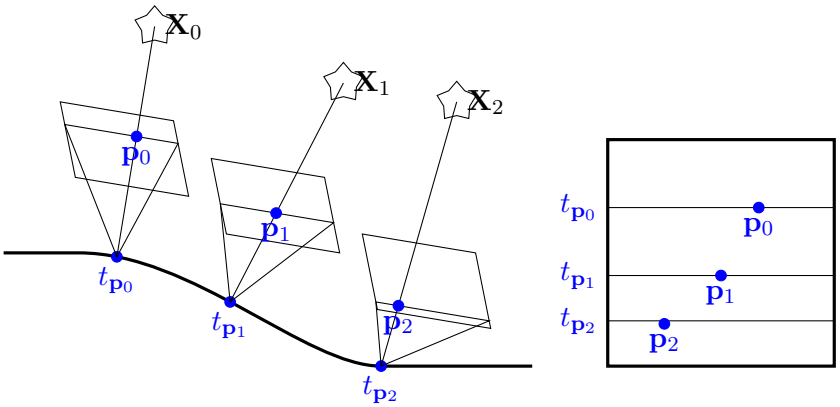


FIGURE 6.1: Rolling shutter problem: a rolling shutter monocular camera moves and sees points at several times/lines in a single frame. Different lines of an image have different poses.

Similar to Chapter 5, we consider $n \geq 2$ jointly linked cameras without assuming that different cameras have common FoV. The cameras are assumed to be frame-accurately (FA) synchronized. We also remind that the intrinsic and extrinsic parameters are initialized using the method in Chapter 5 on the beginning of the videos (the 2k first frames in our experiments) with three approximations: global shutter (GS), central multi-camera and zero subframe residual time offsets. Then the multi-camera GS SfM [Mouragnon+09] (see also Section 2.4) is applied on the whole videos. It generates $m+1$ keyframes and estimates their poses. Let $\mathtt{T}_{\mathrm{M}}^{i}$ be the homogeneous transformation of multi-camera keyframe in the world coordinate system at time t_i ($i \in [0, m]$ is the keyframe number), $\mathtt{T}_{\mathrm{C}}^{j}$ be the homogeneous transformation of the j-th camera in

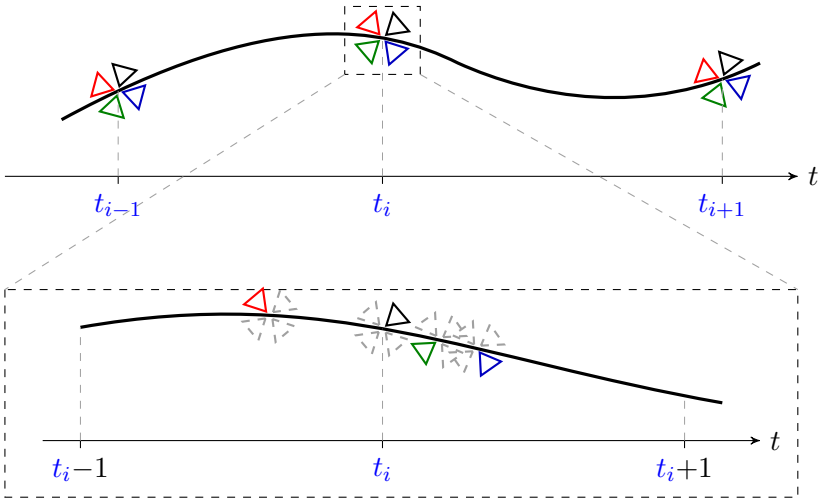


FIGURE 6.2: Inaccurate synchronization problem for a (central) multi-camera: four monocular cameras at the i-th keyframe, which have non-zero time offsets. Different cameras have different poses at the same line.

the multi-camera coordinate system ($j \in [0, n-1]$), see Figure 5.1. Each homogeneous transformation $\mathtt{T}$ is represented with a rotation $\mathtt{R}$ and a translation $\mathbf{t}$ such that

$$\mathtt{T} = \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}. \tag{6.1}$$

In this chapter, our multi-camera BA estimates a subframe-accurate (SFA) synchronization and a line delay coefficient of the RS with usual parameters (intrinsic and extrinsic parameters, multi-camera poses and 3D points). Section 6.2 defines continuous-time parametrization of multi-camera motion. The multi-camera trajectories $M(t)$ are described in Section 6.3 by using the known values of camera motion at time t_i corresponding to the beginning of the keyframes. This is useful to moderate the number of parameters estimated by BA. Section 6.4 explains how to efficiently compute non-closed form forward projections and their derivatives involved in BA. The experiments and conclusion are in Sections 6.6 and 6.7, respectively.

6.2 Parametrization

This section presents our continuous-time parametrization of multi-camera motion: it is defined in Subsection 6.2.1 by the composition of a function M from a time interval to $\mathbb{R}^3 \times \mathbb{R}^k$ and a function $\mathcal{R}$ from $\mathbb{R}^k$ to the set of rotations $\mathbb{SO}(3)$ in $\mathbb{R}^3$. Subsection 6.2.2 describes a keyframe of the multi-camera, where every line has a time that depends on the corresponding camera of the captured line, its v-coordinate and the line delay. The choice of $\mathcal{R}$ is detailed in Subsection 6.2.3.

6.2.1 Parametrization of the multi-camera trajectory

Let $\mathcal{R}$ be a $\mathcal{C}^1$ continuous and surjective function that maps $\mathbb{R}^k$ to $\mathbb{SO}(3)$ (typical values are $k \in \{3, 4\}$). We assume that there is a $\mathcal{C}^3$ continuous function $M : \mathbb{R} \to$

$\mathbb{R}^3 \times \mathbb{R}^k$ that parametrizes the motion of the multi-camera. More precisely, $M(t)^\top = \left(\mathbf{t}_{\mathrm{M}}(t)^\top \mathbf{r}_{\mathrm{M}}(t)^\top\right)$ where $t \in \mathbb{R}$ is the time, $\mathbf{t}_{\mathrm{M}}(t) \in \mathbb{R}^3$ is the translation and $\mathcal{R}(\mathbf{r}_{\mathrm{M}}(t)) \in \mathbb{SO}(3)$ is the rotation of the multi-camera pose. The columns of $\mathcal{R}(\mathbf{r}_{\mathrm{M}}(t))$ and $\mathbf{t}_{\mathrm{M}}(t)$ are the vectors of the multi-camera coordinate system at time t expressed in world coordinates. The choice of $\mathcal{R}$ (including $\mathbf{r}_{\mathrm{M}}$ and k) is detailed in Subsection 6.2.3 for clarity.

Thanks to these notations and assumptions, we will approximate $M(t)$ by using values of M taken at few times $t_0, \ldots, t_m$. Then our model for the camera trajectory not only provides the multi-camera pose at each instant corresponding to each line of a frame, but it also has a moderated number of parameters to be estimated by BA: the vector concatenating all $M(t_i)$ has dimension $(m+1)(3+k)$. Figures 6.1 and 6.2 illustrate trajectory $M(t)$ of a multi-camera defined by four monocular RS cameras having non-zero time offsets. In this chapter, we use shortened notation $\mathbf{m}_i = M(t_i)$.

6.2.2 Time, rolling shutter and synchronization parameters

The i-th keyframe is an image composed of sub-images taken by the monocular cameras. Every line of every sub-image is taken at its own time, which is described now. The 0-th line of the 0-th sub-image in the i-th keyframe is taken at time t_i, assuming that the time exposure of a line is instantaneous [Meingast+05]. Thus $t_{i+1} - t_i$ is a multiple of the inverse of FpS. Since the cameras are RS, the line delay τ is such that the v-th line of the 0-th sub-image in the i-th keyframe is taken at time $t_i + v\tau$. Let $\Delta_j \in \mathbb{R}$ be the sub-frame residual time offset between the j-th video and the 0-th video. Then the 0-th line of the j-th sub-image in the i-th keyframe is taken at time $t_i + \Delta_j$. Since we assume that all cameras have the same FpS and same (and constant) τ, the v-th line of the j-th sub-image in the i-th keyframe is taken at time

$$t(v) = t_i + \Delta_j + v\tau \tag{6.2}$$

as illustrated in Figure 6.3.

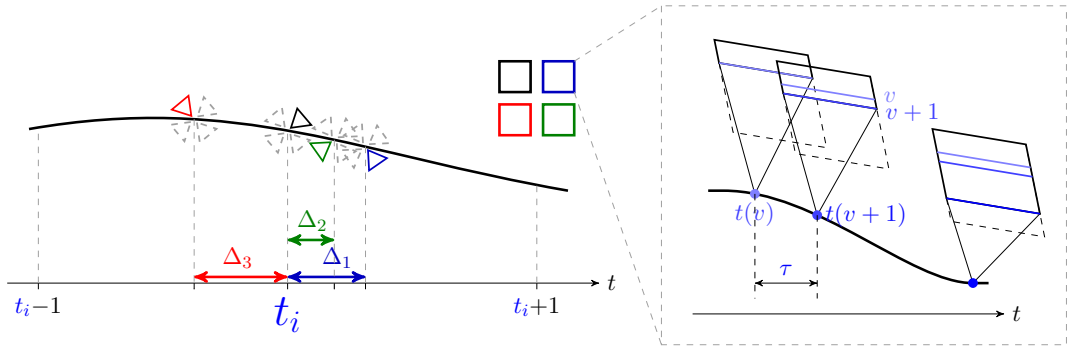


FIGURE 6.3: Time, rolling shutter and synchronization parameters. Left: Δ_j is time offset between the j-th camera and the first camera. Right: τ is line delay between the exposition of two consecutive lines. RS camera moves during the exposure of an image.

6.2.3 Choice of rotation parametrization $\mathcal{R}$

We remind that the rotation representations are discussed in Section 2.2. In the context of the BA in this chapter, we need a rotation parametrization $\mathcal{R}$ that has properties as follows

1. The $\mathcal{C}^1$ continuity of $\mathcal{R}$ is needed by BA for the derivative computation of reprojection errors.
2. Since our BA estimates the multi-camera pose $(\mathbf{t}_{\mathrm{M}}(t_i), \mathcal{R}(\mathbf{r}_{\mathrm{M}}(t_i)))$ using parameters $\mathbf{m}_i = \left(\mathbf{t}_{\mathrm{M}}(t_i)^\top, \mathbf{r}_{\mathrm{M}}(t_i)^\top\right)^\top$, the set of all rotations in a neighborhood of a current estimate of rotation $\mathcal{R}(\mathbf{r}_{\mathrm{M}}(t_i))$ should be reachable by the parametrization $\mathcal{R}$ during every BA iteration [Triggs+00]. Thus, the Jacobian matrix $\partial\mathcal{R}$ of $\mathcal{R}$ should have rank 3 at $\mathbf{r}_{\mathrm{M}}(t_i)$. In other words, $\mathbf{r}_{\mathrm{M}}(t_i)$ should not be a singularity of $\mathcal{R}$.
3. Using a minimal (non-redundant) parametrization $\mathcal{R}$ of the rotations is advisable to limit the number of estimated parameters. Since $\mathcal{R}(\mathbb{R}^k) = \mathbb{SO}(3)$ is a 3D manifold, k should be 3.

All three representations in Section 2.2 satisfy obviously the first property (1). The quaternion representation is free from singularity, hence satisfies the second property (2). $k = 4$, but rotation has only 3 DOFs. This inconvenience can be solved by using minimal parametrization for unit quaternions (see Appendix C.2). The angle-axis representation and Euler angles have only 3 parameters, hence satisfy (3). Unfortunately, these representations have singularities [Hartley+04], [Singla+04]. In our application, this implies that there is always a continuous multi-camera trajectory that crosses a singularity of $\mathcal{R}$. In practice, we believe that no single parametrization of rotation is best for all applications.

Here we consider $\mathcal{R}$ candidates and describe constraints that they induce on a class of multi-camera motions: all yaw motions are possible but pitch and roll are small. Such motions are very common for a helmet-held multi-camera and a user exploring the environment without special objective like grasping at object on the ground (and also for a car-fixed multi-camera).

If we choose the angle-axis representation $\mathbf{v} = \theta\hat{\mathbf{v}}$ (see Section 2.2) as in [Oth+13] and would like to avoid the singularities, the multi-camera should avoid multiple turns on the left (or right) around buildings and avoid straight trajectory segment where $\theta \approx 2\pi\mathbb{Z}^*$.

We also remind the case of Euler angles parametrization (in Section 2.2)

$$\mathcal{E}(\alpha, \beta, \gamma) = \mathtt{R}_z(\gamma)\mathtt{R}_y(\beta)\mathtt{R}_x(\alpha). \tag{6.3}$$

where $\mathtt{R}_x(\alpha), \mathtt{R}_y(\beta)$ and $\mathtt{R}_z(\gamma)$ are the elementary rotations about the axes of coordinate system (defined in Eq.(2.14)). The singularities (α, β, γ) of $\mathtt{R}$ form parallel and equidistant planes of equations $\beta = \pi/2 + \mathbb{Z}\pi$ [Singla+04] (see the more detailed proof in Appendix D.2). In our context, helmet-held and car-fixed multi-camera motions, if we choose $\mathcal{R} = \mathcal{E}$, then the question is how to choose the coordinate systems (both world and multi-camera). If we choose the coordinate systems such that

$\forall i, \mathcal{R}(\mathbf{r}_{\mathrm{M}}(t_i)) \approx \mathtt{R}_x(\alpha_i)$, we are far from the singularities. If we choose the coordinate systems such that $\forall i, \mathcal{R}(\mathbf{r}_{\mathrm{M}}(t_i)) \approx \mathtt{R}_y(\pi/2)\mathtt{R}_x(\alpha_i)$, we are close to the singularities. Inspired by the Euler angle case above, we introduce

$$\mathcal{R}(\alpha, \beta, \gamma) = \mathtt{R}_{\mathtt{A}}\mathtt{R}_z(\gamma)\mathtt{R}_y(\beta)\mathtt{R}_x(\alpha)\mathtt{R}_{\mathtt{B}} \tag{6.4}$$

where rotations $\mathtt{R}_{\mathtt{A}}$ and $\mathtt{R}_{\mathtt{B}}$ do not depend on (α, β, γ). We estimate $\mathtt{R}_{\mathtt{A}}$ and $\mathtt{R}_{\mathtt{B}}$ such that β is close to 0 for all keyframe rotations of the multi-camera trajectory before our BA (technical details in Appendix D.3). Now, the camera motion in our class is far from all singularities.

In this thesis, we experiment on long trajectories in urban scenes. Some of them have many loops. Hence, we choose two global parameterizations for rotations: Euler angles (with our technique to avoid the singularities) and quaternion. In the next section, we explain how to calculate the trajectory $M(t)$ of a multi-camera using these global parameterizations. One important point for unit quaternion parametrization, 4-vector quaternion (i.e. global parameterization, $k = 4$) is used to parametrize the motion of the multi-camera but in BA steps, it is updated using local parameterization (Eq.(C.18)) in Appendix C.2.

6.3 Multi-camera trajectories

In bundle adjustment for RS (and subframe time offset) multi-camera, we need to estimate camera pose at any time. The model of continuous-time multi-camera trajectory depends on the parametrization of camera pose. In this section, we discuss algorithms which approximate for camera trajectory or generate smooth interpolating splines. In addition, it is important that computation of each segment relies on local data only. In such case, a change of values at a particular knot does not imply recalculation of the whole curve, but only neighbor segments.

The following subsections focus on the model of multi-camera trajectory and three different approaches are introduced. The first (Subsection 6.3.1) concerns the approximation using Taylor's expansion for the Euclidean space. The second (Subsection 6.3.2) presents the algorithms for unit quaternion in Subsection 6.3.2.1 and Lie groups in Subsection 6.3.2.2.

Through the section, $\mathbb{M}$ denotes a manifold ($\mathbb{R}^k$ or $\mathbb{SO}(3)$ or $\mathbb{SE}(3)$ in our work) and $\mathbb{T}_m\mathbb{M}$ is the tangent space to $\mathbb{M}$, at a point $\mathbf{m} \in \mathbb{M}$ (see Appendix F.1). The general interpolation problem to be studied in this section has the following statement:

Problem 1. Given $m+1$ distinct points $\mathbf{m}_0, \mathbf{m}_1, \ldots, \mathbf{m}_m$ in the manifold $\mathbb{M}$, $m+1$ vectors $\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_m$ tangent to $\mathbb{M}$ at points $\mathbf{m}_0, \mathbf{m}_1, \ldots, \mathbf{m}_m$ respectively, and a partition $0 = t_0 < t_1 < \ldots < t_m = T$ of the time interval $[0, T]$, generate a $\mathcal{C}^l$ continuous smooth curve ($l \geq 1$) $M : [0, T] \subset \mathbb{R} \to \mathbb{M}$ on $\mathbb{M}$, which satisfies the following conditions:

$$M(t_i) = \mathbf{m}_i, \quad \dot{M}(t_i) = \mathbf{v}_i, \qquad 0 \leq i \leq m. \tag{6.5}$$

6.3.1 Euclidean space

We start with the case when $\mathbb{M}$ is the k-dimensional Euclidean space $\mathbb{R}^k$. We can use for the case: Euler's angles ($\mathbb{R}^3$) or quaternions ($\mathbb{R}^4$) and translation ($\mathbb{R}^3$). Let $\delta = \max_i(t_{i+1} - t_i)$. Thanks to the $\mathcal{C}^3$ continuity of M and Taylor's expansions of M at t_i, we explicit two approximations $M_1(t)$ and $M_2(t)$ of $M(t)$ in the neighborhood of t_i as functions of $\mathbf{m}_{i-1}, \mathbf{m}_i$ and $\mathbf{m}_{i+1}$ with remainders expressed in terms of δ and $|t - t_i|$. By neglecting these remainders, we compute $M(t)$ for the v-th line of the j-th camera/sub-image in the i-th keyframe using $t = t_i + \Delta_j + \tau(0 \quad 1)\mathbf{p}$ (Subsection 6.2.2) during our BA. We remind that $\mathbf{m}_i = M(t_i)$.

6.3.1.1 Linear approximation M_1 of M

We have Taylor's linear expansion

$$M(t) = \mathbf{m}_i + (t - t_i)\dot{M}(t_i) + \mathcal{O}(|t - t_i|^2) \tag{6.6}$$

and express derivative $\dot{M}(t_i)$ as a function of $\mathbf{m}_{i-1}, \mathbf{m}_i$ and $\mathbf{m}_{i+1}$ using the following lemma.

Lemma 6.1. *Let vectors $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^k$, strictly positive reals a, b and function*

$$D_1(\mathbf{x}, \mathbf{y}, \mathbf{z}, a, b) = -\frac{a\mathbf{x}}{b(a+b)} + \frac{(a-b)\mathbf{y}}{ab} + \frac{b\mathbf{z}}{a(a+b)}. \tag{6.7}$$

If $M : \mathbb{R} \to \mathbb{R}^k$ is a $\mathcal{C}^3$ continuous function and $t \in \mathbb{R}$,

$$\dot{M}(t) = D_1(M(t-b), M(t), M(t+a), a, b) + \mathcal{O}(a^2 + b^2). \tag{6.8}$$

Proof. Since M is $\mathcal{C}^3$ continuous,

$$M(t+a) = M(t) + a\dot{M}(t) + \frac{a^2}{2}\ddot{M}(t) + \mathcal{O}(a^3), \tag{6.9}$$

$$M(t-b) = M(t) - b\dot{M}(t) + \frac{b^2}{2}\ddot{M}(t) + \mathcal{O}(b^3). \tag{6.10}$$

We eliminate $\ddot{M}(t)$ by summing $\frac{b}{a}$ Eq.(6.9) $-\frac{a}{b}$ Eq.(6.10):

$$\frac{b}{a}M(t+a) - \frac{a}{b}M(t-b) = \left(\frac{b}{a} - \frac{a}{b}\right)M(t) + (b+a)\dot{M}(t) + b\mathcal{O}(a^2) + a\mathcal{O}(b^2). \tag{6.11}$$

Since $a > 0$ and $b > 0$,

$$\dot{M}(t) = \frac{1}{a+b}\left(\frac{b}{a}M(t+a) - \frac{a}{b}M(t-b) + \left(\frac{a}{b} - \frac{b}{a}\right)M(t)\right) + \mathcal{O}(a^2 + b^2). \tag{6.12}$$

□

Thanks to Lemma 6.1 with $t = t_i$, $a = t_{i+1} - t_i$, $b = t_i - t_{i-1}$, we have

$$\dot{M}(t_i) = D_1(\mathbf{m}_{i-1}, \mathbf{m}_i, \mathbf{m}_{i+1}, t_{i+1} - t_i, t_i - t_{i-1}) + \mathcal{O}(\delta^2). \tag{6.13}$$

The value of $M(t)$ is approximated by neglecting all remainders expressed by "$\mathcal{O}$" above and using 3 neighboring control points $\mathbf{m}_{i-1}, \mathbf{m}_i$ and $\mathbf{m}_{i+1}$ as illustrated in Figure 6.4. For $0 < i < m$ and $t \approx t_i$,

$$M(t) = \mathbf{m}_i + (t - t_i)D_1(\mathbf{m}_{i-1}, \mathbf{m}_i, \mathbf{m}_{i+1}, t_{i+1} - t_i, t_i - t_{i-1}). \tag{6.14}$$

For $i = 0$ (similarly for $i = m$), we use $M(t) = \mathbf{m}_0 + \dfrac{t - t_0}{t_1 - t_0}(\mathbf{m}_1 - \mathbf{m}_0)$.

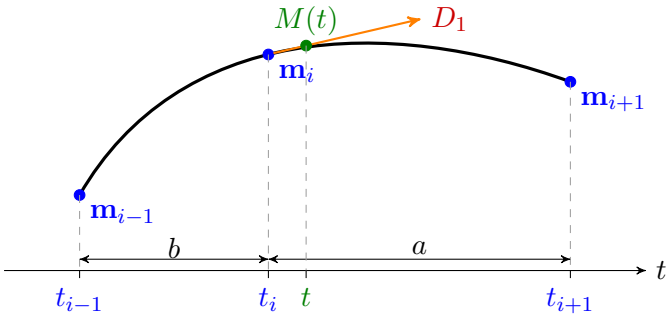


FIGURE 6.4: Linear approximation M_1 of M.

6.3.1.2 Quadratic approximation M_2 of M

Similarly, we have Taylor's quadratic expansion of M at t_i

$$M(t) = \mathbf{m}_i + (t - t_i)\dot{M}(t_i) + \frac{(t - t_i)^2}{2}\ddot{M}(t_i) + \mathcal{O}(|t - t_i|^3) \tag{6.15}$$

and express derivative $\ddot{M}(t_i)$ as a function of $\mathbf{m}_{i-1}, \mathbf{m}_i$ and $\mathbf{m}_{i+1}$.

Lemma 6.2. *Let vectors $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^k$, strictly positive reals a, b and function*

$$D_2(\mathbf{x}, \mathbf{y}, \mathbf{z}, a, b) = \frac{2\mathbf{x}}{b(a+b)} - \frac{2\mathbf{y}}{ab} + \frac{2\mathbf{z}}{a(a+b)}. \tag{6.16}$$

If $M : \mathbb{R} \to \mathbb{R}^k$ is a $\mathcal{C}^3$ continuous function and $t \in \mathbb{R}$,

$$\ddot{M}(t) = D_2(M(t-b), M(t), M(t+a), a, b) + \mathcal{O}(a+b). \tag{6.17}$$

Proof. We eliminate $\dot{M}(t)$ by summing b Eq.(6.9) $+a$ Eq.(6.10):

$$bM(t+a) + aM(t-b) = (a+b)M(t) + \frac{ab}{2}(a+b)\ddot{M}(t) + \mathcal{O}(ba^3 + ab^3). \tag{6.18}$$

Since $a > 0$ and $b > 0$,

$$\ddot{M}(t) = \frac{2}{a(a+b)}M(t+a) + \frac{2}{b(a+b)}M(t-b) - \frac{2}{ab}M(t) + \mathcal{O}(a+b). \tag{6.19}$$

□

Using Lemma 6.2 with $t = t_i$, $a = t_{i+1} - t_i$, $b = t_i - t_{i-1}$, we obtain

$$\ddot{M}(t_i) = D_2(\mathbf{m}_{i-1}, \mathbf{m}_i, \mathbf{m}_{i+1}, t_{i+1} - t_i, t_i - t_{i-1}) + \mathcal{O}(\delta). \tag{6.20}$$

We approximate $M(t)$ from the $\mathbf{m}_i$ by neglecting all remainders expressed by "$\mathcal{O}$" in Eq.(6.13) and Eq.(6.20). The value of $M(t)$ is approximated by using 3 neighboring control points $\mathbf{m}_{i-1}, \mathbf{m}_i$ and $\mathbf{m}_{i+1}$. For $0 < i < m$ and $t \approx t_i$,

$$\begin{aligned} M(t) = \quad & \mathbf{m}_i + (t - t_i)D_1(\mathbf{m}_{i-1}, \mathbf{m}_i, \mathbf{m}_{i+1}, t_{i+1} - t_i, t_i - t_{i-1}) \\ & + \frac{(t-t_i)^2}{2} D_2(\mathbf{m}_{i-1}, \mathbf{m}_i, \mathbf{m}_{i+1}, t_{i+1} - t_i, t_i - t_{i-1}). \end{aligned} \tag{6.21}$$

For $i = 0$ (similarly for $i = m$), we use $M(t) = \mathbf{m}_0 + \dfrac{t - t_0}{t_1 - t_0}(\mathbf{m}_1 - \mathbf{m}_0)$.

6.3.1.3 Checking the remainders

We remind that the remainders of our Taylor developments are $\mathcal{O}(|t-t_i|^2)$ in Eq.(6.6), $\mathcal{O}(\delta^2)$ in Eq.(6.13), $\mathcal{O}(|t - t_i|^3)$ in Eq.(6.15) and $\mathcal{O}(\delta)$ in Eq.(6.20). Here we only check that the remainders above have intuitive behavior: they converge to 0 if both FpS and keyframe density increase. More precisely, if the density of control points (keyframes in the videos) increases (and if there is not stop-and-go in video acquisition), δ decreases. Furthermore, $|v\tau| \leq 1/\text{FpS}$ and the FA synchronization provides $|\Delta_j| \leq 1/\text{FpS}$. Thus $|t - t_i| = |\Delta_j + v\tau|$ decreases if the FpS increases.

6.3.2 Spherical interpolation

In this subsection, we describe two spherical interpolations: *Squad* for unit quaternions and spline on $\mathbb{SE}(3)$.

6.3.2.1 Spherical and quadrangle (*Squad*)

The basic quaternion mathematics are stated in Appendix E (more details in [Dam+98]). The set of unit quaternions is denoted $\mathbb{H}_1$. Let $\mathbf{q}_1, \mathbf{q}_2 \in \mathbb{H}_1$ and $h \in [0, 1]$.

Definition 6.3. The **s**pherical **l**inear int**erp**olation (*Slerp*) has four equivalent definitions [Dam+98]:

$$Slerp(\mathbf{q}_1, \mathbf{q}_2, h) = \mathbf{q}_1(\mathbf{q}_1^*\mathbf{q}_2)^h \tag{6.22a}$$

$$Slerp(\mathbf{q}_1, \mathbf{q}_2, h) = (\mathbf{q}_1\mathbf{q}_2^*)^{1-h}\mathbf{q}_2 \tag{6.22b}$$

$$Slerp(\mathbf{q}_1, \mathbf{q}_2, h) = (\mathbf{q}_2\mathbf{q}_1^*)^h\mathbf{q}_1 \tag{6.22c}$$

$$Slerp(\mathbf{q}_1, \mathbf{q}_2, h) = \mathbf{q}_2(\mathbf{q}_2^*\mathbf{q}_1)^{1-h}. \tag{6.22d}$$

Note that $Slerp(\mathbf{q}_1, \mathbf{q}_2, h) = Slerp(\mathbf{q}_2, \mathbf{q}_1, 1-h)$. The interpolation curve for *Slerp* forms a great arc and shortest on the unit quaternion sphere between $\mathbf{q}_1$ and $\mathbf{q}_2$. In differential geometry terms, the great arc is a geodesic - corresponding to a straight line in $\mathbb{R}^3$. Furthermore, *Slerp* has constant angular velocity. So *Slerp* is a convenient choice for interpolating between two rotations.

Slerp is similar to linear interpolation in $\mathbb{R}^2$ between two points. If we take an ordered set of points and generate values somewhere between consecutive points using linear interpolation, we come up against problems: the curve is not smooth at the control points and the velocity is not constant and not continuous at the control points. The same problems occur if we interpolate between a series of rotations using *Slerp*. When interpolating between a series of control points in a plane, different kinds of cubic curves such as Bézier curve can be constructed quite simply. Bézier curves are the result of linear interpolations. Let's start with the simple explanation for a quadratic Bézier curve from three points. Three points give us two lines. Linear interpolations over these lines gives us two points, between which we can again perform linear interpolation, yielding a single point. All points that we can form in this way taken together form our quadratic Bézier curve. Inspired by the construction of Bézier curve, spherical cubic equivalent of a Bézier curve is developed. This interpolation curve is called *Squad* (**s**pherical and **quad**rangle) [Dam+98] for uniform distribution of control points. Figure 6.5 illustrates a simple linear interpolation, a Bézier curve, *Slerp* and *Squad*. For *non-uniform* distribution of control points,

Definition 6.4. The non-uniform spherical spline quaternion interpolation (*Squad*) is defined by

$$Squad(\mathbf{q}_i, \mathbf{q}_{i+1}, \mathbf{a}_i, \mathbf{a}_{i+1}, t) = Slerp\left(Slerp(\mathbf{q}_i, \mathbf{q}_{i+1}, h), Slerp(\mathbf{a}_i, \mathbf{a}_{i+1}, h), 2h(1-h)\right) \tag{6.23}$$

with $t_i \leq t \leq t_{i+1}, \delta_i = t_{i+1} - t_i, h = \dfrac{t - t_i}{\delta_i}$,

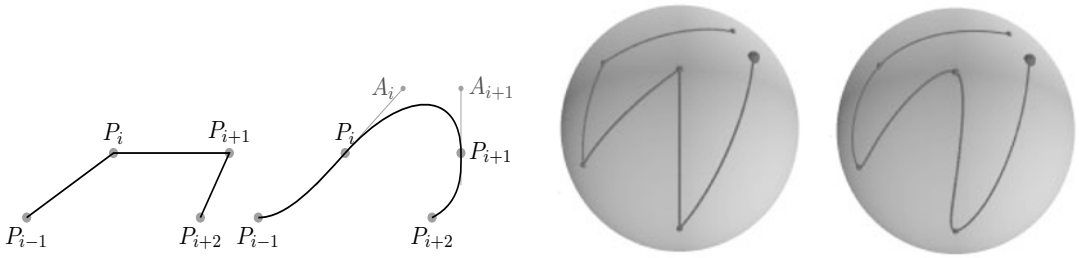
$$\mathbf{a}_i = \mathbf{q}_i \exp\left(-\frac{\delta_i \log(\mathbf{q}_i^{-1}\mathbf{q}_{i-1}) + \delta_{i-1}\log(\mathbf{q}_i^{-1}\mathbf{q}_{i+1})}{2(\delta_i + \delta_{i-1})}\right). \tag{6.24}$$

The resulting expressions for *Squad* involves the spherical linear interpolation in Def.6.3. The interpolation curve between $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$ is solely determined from the positions of auxiliary points $\mathbf{a}_i, \mathbf{a}_{i+1}$ defined from control points $\mathbf{q}_{i-1}, \mathbf{q}_i, \mathbf{q}_{i+1}$ and $\mathbf{q}_{i+2}$. The tangent at $\mathbf{q}_i$ is defined by the vector $\mathbf{a}_i - \mathbf{q}_i$ and the tangent at $\mathbf{q}_{i+1}$ is defined by the vector $\mathbf{a}_{i+1} - \mathbf{q}_{i+1}$. So, *Squad* interpolates from 4 neighboring control points $\mathbf{q}_{i-1}, \mathbf{q}_i, \mathbf{q}_{i+1}$ and $\mathbf{q}_{i+2}$ for $t_i \leq t \leq t_{i+1}$.

Proposition 6.5. *Squad* $\in \mathcal{C}^1$.

Proof. See [Dam+98]. □

The expression for *Squad* in Eq.(6.23) yields an interpolation curve for a series of quaternions $\mathbf{m}_0, \ldots, \mathbf{m}_m$ (in Problem 1). The value of $M(t)$ is evaluated from 4 neighboring control points for every t. The expression is not defined in the first and last interval since $\mathbf{m}_{-1}$ appears in the expression for $\mathbf{a}_0$ and $\mathbf{m}_{m+1}$ appears in the expression



(a) Linear interp. and 3rd-order Bézier curve (b) *Slerp* and *Squad*

FIGURE 6.5: Interpolation curve for 3rd-order Bézier curve, *Slerp* and *Squad* (thanks to [Dam+98]). a) Linear interpolation between a series of points and interpolation between the points P_i and P_{i+1} with a 3rd-order Bézier curve where the tangent at the control points is defined by auxiliary points A_i and A_{i+1}, respectively; b) *Slerp* and *Squad* between 6 points.

for $\mathbf{a}_m$. Therefore, it is necessary to define sound values for $\mathbf{a}_0$ and $\mathbf{a}_m$. The simplest solution is to define $\mathbf{a}_0 \equiv \mathbf{m}_0$ and $\mathbf{a}_m \equiv \mathbf{m}_m$. As another solution, we choose the interpolation curve of *Slerp* for the first and last interval. For translation parameters, we use the approximation as in Subsection 6.3.1.

6.3.2.2 Spline on Lie Group $\mathbb{SE}(3)$

The set of matrix

$$\mathbb{SE}(3) = \left\{ \mathtt{T} \in \mathbb{R}^{4\times 4} \,\middle|\, \mathtt{T} = \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \mathtt{R} \in \mathbb{SO}(3), \mathbf{t} \in \mathbb{R}^3 \right\} \tag{6.25}$$

forms the matrix group under the operation of matrix multiplication. The concepts of Lie Group $\mathbb{SE}(3)$ are more detailed in Appendix F. In this section, we present a smooth spline generation on Lie Group $\mathbb{SE}(3)$. So, in this section, we use the control point $\mathtt{T}_i$ instead of $\mathbf{m}_i$, tangent $\mathtt{V}_i$ in place of $\mathbf{v}_i$ and matrix $\mathtt{T}(t)$ in place of its parameter vector $M(t)$ in Problem 1.

[Jakubiak+06] proposes the algorithm of spline construction for k-dimensional Euclidean space $\mathbb{R}^k$ and generalizes it to manifold such as a Lie group or a sphere. For our matrix Lie group $\mathbb{SE}(3)$ with Lie algebra $\mathfrak{se}(3)$, any vector tangent to $\mathbb{SE}(3)$ at a point $\mathtt{T}$ is of the form $\mathbf{V}\mathtt{T}$, for some generators $\mathbf{V} \in \mathfrak{se}(3)$. Each pair $(\mathtt{T}_i, \mathbf{V}_i) \in \mathbb{SE}(3) \times \mathfrak{se}(3)$ determines a unique geodesic $t \mapsto x(t)$, which passes through $\mathtt{T}_i$ at $t = t_i$ with velocity equal to $\mathbf{V}_i\mathtt{T}_i$. This geodesic is defined in terms of the exponential mapping by $x(t) = e^{(t-t_i)\mathbf{V}_i}\mathtt{T}_i$.

For $t_i \le t \le t_{i+1}$, let $\delta_i = t_{i+1} - t_i$, $h = \dfrac{t - t_i}{\delta_i} \in [0, 1]$. First, the left and right components of the spline segment $t \mapsto \mathtt{T}(t)$ which satisfies the boundary conditions:

$$\begin{aligned} \mathtt{T}(t_i) &= \mathtt{T}_i, & \mathtt{T}(t_{i+1}) &= \mathtt{T}_{i+1}, \\ \dot{\mathtt{T}}(t_i) &= \mathtt{V}_i = \mathbf{V}_i\mathtt{T}_i, & \dot{\mathtt{T}}(t_{i+1}) &= \mathtt{V}_{i+1} = \mathbf{V}_{i+1}\mathtt{T}_{i+1} \end{aligned} \tag{6.26}$$

are given respectively by

$$L_i(t) = e^{(t-t_i)\mathbf{V}_i}\mathtt{T}_i \quad \text{and} \quad R_i(t) = e^{(t-t_{i+1})\mathbf{V}_{i+1}}\mathtt{T}_{i+1}. \tag{6.27}$$

Second, the authors introduce a $\mathcal{C}^l$-smooth real-valued function $\phi : [0,1] \to [0,1]$ satisfying

$$\begin{aligned} \phi(0) &= 0, \quad \phi(1) = 1, \\ \phi^{(j)}(0) &= 0, \quad \phi^{(j)}(1) = 0, \qquad j = 1, 2, \ldots, l-1, \quad (\text{for } l > 1), \end{aligned} \tag{6.28}$$

and the curve $t \mapsto \mathtt{T}(t)$, defined by

$$\mathtt{T}(t) = e^{\phi(h)\mathbf{B}_i(t)} L_i(t) \tag{6.29}$$

where

$$\mathbf{B}_i(t) = \log\left(R_i(t) L_i^{-1}(t)\right) \tag{6.30}$$

satisfies the required boundary conditions (Eq.(6.26)) according to [Jakubiak+06, Theorem 4.1]. We remind that $h = \dfrac{t - t_i}{t_{i+1} - t_i} \in [0,1]$.

The smoothness properties of the spline Eq.(6.29) depend highly on the choice of the function ϕ. For a fixed l, the resulting spline using the function ϕ given in [Jakubiak+06, Lemma 3.2], is $\mathcal{C}^l$-smooth. Thus ϕ is called a smoothing function for the spline. Given l, the coefficients of the smoothing function ϕ can be obtained using [Jakubiak+06, Lemma 3.2]. For small values of l,

$$\begin{aligned} l = 1, &\quad \phi(h) = 3h^2 - 2h^3, \\ l = 2, &\quad \phi(h) = 10h^3 - 15h^4 + 6h^5, \\ l = 3, &\quad \phi(h) = 35h^4 - 84h^5 + 70h^6 - 20h^7, \\ l = 4, &\quad \phi(h) = 126h^5 - 420h^6 + 540h^7 - 315h^8 + 70h^9. \end{aligned} \tag{6.31}$$

The coefficients are stored in a look-up table.

Proposition 6.6. *The curve $t \mapsto \mathtt{T}(t)$ defined by Eqs.(6.27,6.29,6.30) with $t_i \leq t \leq t_{i+1}$ is a $\mathcal{C}^1$ continuous function.*

Proof. See [Jakubiak+06]. □

The expression Eq.(6.29) yields an interpolation curve for a series of poses $\mathtt{T}_0, \ldots, \mathtt{T}_m$ (in Problem 1). In our BA problem, we do not have velocity $\mathtt{V}_i$ at control point $\mathtt{T}_i$, in practice. Therefore, it is necessary to define values for $\mathtt{V}_i = \mathbf{V}_i \mathtt{T}_i$. The simplest solution is to define:

$$\mathbf{V}_i = \frac{\log\left(\mathtt{T}_{i+1} \mathtt{T}_{i-1}^{-1}\right)}{\delta_{i-1} + \delta_i}. \tag{6.32}$$

Thus the value of $\mathtt{T}(t)$ (hence its parameter vector $M(t)$) is evaluated from 4 neighboring control points $\mathtt{T}_{i-1}$, $\mathtt{T}_i$, $\mathtt{T}_{i+1}$ and $\mathtt{T}_{i+2}$, for $t_i \leq t \leq t_{i+1}$. Moreover, the expression is not defined in the first and last interval since $\mathtt{T}_{-1}$ appears in the expression for $\mathtt{V}_0$ and $\mathtt{T}_{m+1}$ appears in the expression for $\mathtt{V}_m$. For $t_0 \leq t \leq t_1$, we choose the approximation following

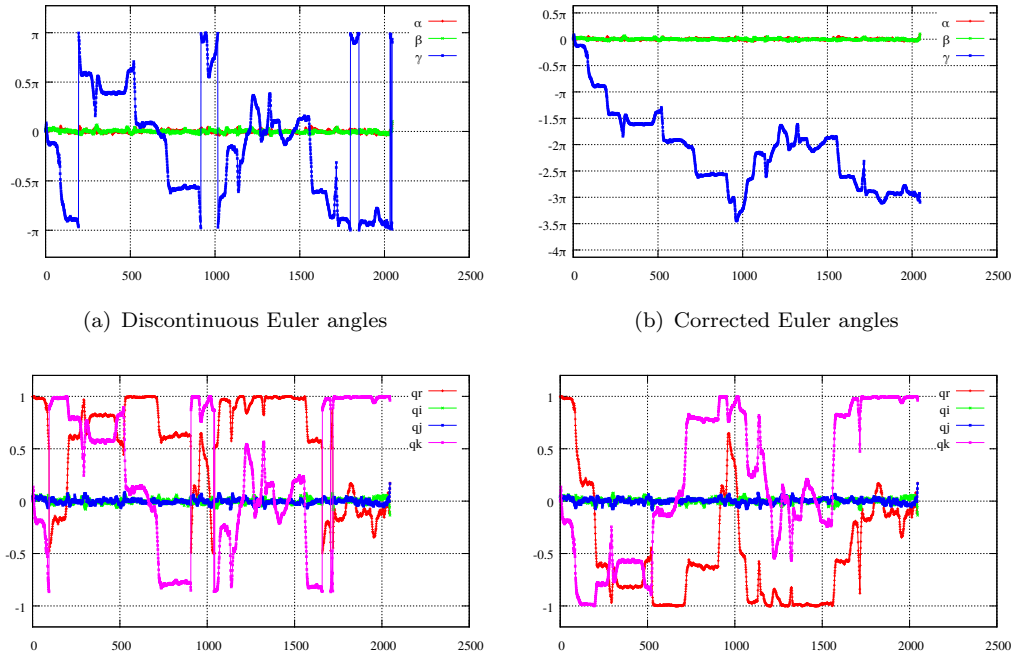
$$\mathtt{T}(t) = \exp((t - t_0)\mathbf{V}_0)\mathtt{T}_0 \tag{6.33}$$

where $\mathbf{V}_0 = \frac{1}{\delta_0} \log(\mathtt{T}_1 \mathtt{T}_0^{-1})$. It is similar for the last interval.

6.3.3 Enforce continuity of M

Remind that $t \mapsto M(t)$ should be continuous ($\mathcal{C}^3$ continuous). Furthermore, $t_{i+1} - t_i$ is small thanks to the keyframe sampling in SfM (see Section 2.4.1). Since our rotation parameterizations are not injective (both Euler-based and quaternion), we choose $M(t_i)$ such that $M(t_{i+1}) - M(t_i)$ is as small as possible.

For Euler angle representation, we do an assumption on the camera motion to keep away from the singularities (Subsection 6.2.3). Note that in Chapter 5, we use local Euler angle in order to avoid singularities. By contrast, we consider global Euler angles for the continuous camera motion $M(t)$ in this chapter. The conversion from rotation matrix to Euler angle in Appendix D.1 yields a 3-vector (α, β, γ) where $-\pi \leq \alpha \leq \pi, -\pi/2 \leq \beta \leq \pi/2, -\pi \leq \gamma \leq \pi$ (Note that, there are two solutions; we only choose one and the same for all rotation matrices). Furthermore, β is close to 0 thanks to Subsection 6.2.3. Figure 6.6(a) gives an example. We clearly see discontinuities of Euler angles, although they are assumed to be a continuous function with respect to time. Here, we propose a simple correction. We take γ for example, this is similar for α. In fact, angles γ and $\gamma + d_\gamma 2\pi$ ($d_\gamma \in \mathbb{Z}$) represent the same rotation using $\mathtt{R}_z$ (Eq.(2.14)). Firstly, we set $d_\gamma = 0$. Then we do a loop on i ($0 < i \leq m$). If $\gamma_{i-1} - \gamma_i > \pi$ then d_γ increases by 1. If $\gamma_i - \gamma_{i-1} > \pi$ then d_γ decreases by 1. Now γ_i is replaced by $\gamma_i + d_\gamma 2\pi$ and i increases by 1. Figure 6.6(b) illustrates this correction. This preprocessing is necessary to use the approximations in Eq.(6.14) and Eq.(6.21).



(a) Discontinuous Euler angles

(b) Corrected Euler angles

(c) Discontinuous quaternions

(d) Corrected quaternions

FIGURE 6.6: Example for rotation representation parameters (for BC1). x-axis are the keyframe numbers, y-axis are Euler angles (top) or quaternions (bottom).

Quaternion representation is a singularity-free but redundant parametrization (it has 4 parameters instead of the minimum 3). Note that, the multi-camera motion in Section 6.3 requires unit-length constraint of quaternions at control points. Moreover, according to Appendix E, $\mathbf{q}_i$ and $-\mathbf{q}_i$ perform the same rotation. However, they could

possibly give a significant difference when we apply the approximations in Eq.(6.14), Eq.(6.21) and *Squad* interpolation in Eq.(6.23), i.e. the interpolation between $\mathbf{q}_{i-1}$ and $-\mathbf{q}_i$ yield a shorter interpolation path than the interpolation between $\mathbf{q}_{i-1}$ and $\mathbf{q}_i$ (see Figure 6.6(c)). A simple correction is to verify the angle θ between two quaternion $\mathbf{q}_{i-1}$ and $\mathbf{q}_i$ using quaternion inner product (Appendix E, Eq.(E.6)). If θ is an acute angle, interpolation path is shortest. Otherwise, we replace $\mathbf{q}_i$ by $-\mathbf{q}_i$. Like Euler angles case, this preprocessing is necessary to obtain a continuous camera motion model.

6.4 Non-closed form image projections

Since our BA minimizes the sum of squared modulus of reprojection error for every inlier, this section describes the computation of a reprojection error for 3D point $\mathbf{X}^l \in \mathbb{R}^4$ (in the homogeneous world coordinates) and its inlier observation $\tilde{\mathbf{p}} \in \mathbb{R}^2$ in the j-th sub-image of the i-th keyframe.

First, we introduce notations. Let $\mathbf{p} \in \mathbb{R}^2$ be the projection of $\mathbf{X}^l$ in the j-th sub-image of the i-th keyframe. The reprojection error is

$$\boldsymbol{\epsilon} = \mathbf{p} - \tilde{\mathbf{p}}. \tag{6.34}$$

Let $(\mathtt{R}_{\mathrm{C}}^j, \mathbf{t}_{\mathrm{C}}^j)$ be the pose of the j-th camera in the multi-camera frame and $\mathbf{m}_i(\mathtt{R}_{\mathrm{M}}^i, \mathbf{t}_{\mathrm{M}}^i)$ be the parameter vector of the pose of the i-th keyframe in the word coordinate system. Let $p_j : \mathbb{R}^4 \setminus \{\mathbf{0}\} \to \mathbb{R}^2$ be the projection function of the j-th camera. We assume that $p_j, \mathtt{R}_j, \mathbf{t}_j$ are constant. The acquisition times of $\mathbf{p} = (u, v)^\top$ and $\tilde{\mathbf{p}} = (\tilde{u}, \tilde{v})^\top$ are

$$t_{\mathbf{p}} = t_i + \Delta_j + v\tau \quad \text{and} \quad t_{\tilde{\mathbf{p}}} = t_i + \Delta_j + \tilde{v}\tau. \tag{6.35}$$

Second, we detail the relation between $\mathbf{p}$ and $\mathbf{X}^l$. Both $\mathbf{r}_{\mathrm{M}}(t_{\mathbf{p}})$ and $\mathbf{t}_{\mathrm{M}}(t_{\mathbf{p}})$, i.e. $M(t_{\mathbf{p}})$, are defined by one equation chosen among Eq.(6.14), Eq.(6.21), Eq.(6.23) and Eq.(6.29) (see Table 6.1) using index i of the keyframe and $t = t_{\mathbf{p}}$. The homogeneous coordinates of $\mathbf{X}^l$ in the multi-camera coordinate system is

$$\mathbf{X}_{\mathrm{M}} = \begin{bmatrix} \mathcal{R}^\top(\mathbf{r}_{\mathrm{M}}(t_{\mathbf{p}})) & -\mathcal{R}^\top(\mathbf{r}_{\mathrm{M}}(t_{\mathbf{p}}))\mathbf{t}_{\mathrm{M}}(t_{\mathbf{p}}) \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{X}^l. \tag{6.36}$$

The homogeneous coordinates of $\mathbf{X}^l$ in the j-th camera coordinate system and the projection of $\mathbf{X}^l$ are

$$\mathbf{X}_{\mathrm{C}} = \begin{bmatrix} (\mathtt{R}_{\mathrm{C}}^j)^\top & -(\mathtt{R}_{\mathrm{C}}^j)^\top \mathbf{t}_{\mathrm{C}}^j \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{X}_{\mathrm{M}} \quad \text{and} \quad \mathbf{p} = p_j(\mathbf{X}_{\mathrm{C}}). \tag{6.37}$$

We see that $\mathbf{p}$ needs the computation of $\mathbf{X}_{\mathrm{M}}$ (Eq.(6.37)), which in turn needs the computation of (the v coordinate of) $\mathbf{p}$ in Eq.(6.36). This is a *chicken-egg* problem.

We should also compute derivatives of projection $\mathbf{p}$ with respect to a vector $\boldsymbol{\theta}$ of parameters optimized by BA (among τ, Δ_j, intrinsic and distortion parameters, extrinsic parameters $(\mathtt{R}_{\mathrm{C}}^j, \mathbf{t}_{\mathrm{C}}^j)$, camera motion $\mathbf{m}_i$, 3D point $\mathbf{X}^l$) although $\mathbf{p}$ does not have a closed-form expression from them. This is needed by BA.

6.4.1 General case

We know an approximate value $\tilde{\mathbf{p}}$ of $\mathbf{p}$ (since $\tilde{\mathbf{p}}$ is an inlier detected in an image), a $\mathcal{C}^1$ continuous function $g(\mathbf{z}, \boldsymbol{\theta})$ from $\mathbb{R}^2 \times \mathbb{R}^p$ to $\mathbb{R}^2$ such that $\mathbf{p}$ is the solution $\mathbf{z}$ of $g(\mathbf{z}, \boldsymbol{\theta}) = \mathbf{0}$, and the current value $\boldsymbol{\theta}_0$ of $\boldsymbol{\theta}$ (provided by initialization or previous iteration of BA). First $\mathbf{p}$ is estimated by non-linear least-squares minimizing $\mathbf{z} \mapsto ||g(\mathbf{z}, \boldsymbol{\theta}_0)||^2$. In practice, we use the iterative Gauss-Newton's method starting from $\mathbf{z} = \tilde{\mathbf{p}}$ with no more than 5 iterations (Newton's method can also be used). Then the implicit function Theorem implies that we locally have a $\mathcal{C}^1$ continuous function ψ such that $\mathbf{p} = \psi(\boldsymbol{\theta})$ if $\det \frac{\partial g}{\partial \mathbf{z}} \neq 0$. By differentiating $g(\psi(\boldsymbol{\theta}), \boldsymbol{\theta}) = \mathbf{0}$ using the Chain rule, we obtain

$$\frac{\partial \mathbf{p}}{\partial \boldsymbol{\theta}} = \frac{\partial \psi}{\partial \boldsymbol{\theta}} = -\left(\frac{\partial g}{\partial \mathbf{z}}\right)^{-1} \frac{\partial g}{\partial \boldsymbol{\theta}}. \tag{6.38}$$

6.4.2 Approximate solution

The chicken-egg problem above is solved thanks to an approximation in [Klingner+13]: $t_{\mathbf{p}}$ is replaced by $t_{\tilde{\mathbf{p}}}$ in Eq.(6.36), i.e. the authors assume that the multi-camera pose is the same at times $t_{\tilde{\mathbf{p}}}$ and $t_{\mathbf{p}}$. We think that this is acceptable since $|t_{\tilde{\mathbf{p}}} - t_{\mathbf{p}}| \leq \tau ||\mathbf{p} - \tilde{\mathbf{p}}||$ and the magnitude order of τ is 10^{-5} s/pixel and $\tilde{\mathbf{p}}$ is an inlier (i.e. $||\mathbf{p} - \tilde{\mathbf{p}}|| \leq 4$ pixels).

By using this approximation, the acquisition time of $\mathbf{p}$ is

$$t(\Delta_j, \tau, \mathbf{p}) \approx t_i + \Delta_j + \tau(0 \quad 1)\tilde{\mathbf{p}} \tag{6.39}$$

where $\tilde{\mathbf{p}}$ is known. Let $\boldsymbol{\theta}_j$ be the vector concatenating the intrinsic/distortion parameters and the pose of the j-th camera in the multi-camera coordinate system. If the camera motion $M(\mathbf{m}_{i-1}, \mathbf{m}_i, \mathbf{m}_{i+1}, t)$ is approximated by Eq.(6.14) or Eq.(6.21), the forward projection is a function

$$\mathbf{p} = p_j\left(\boldsymbol{\theta}_j, M\left(\mathbf{m}_{i-1}, \mathbf{m}_i, \mathbf{m}_{i+1}, t(\Delta_j, \tau)\right), \mathbf{X}\right). \tag{6.40}$$

In the same way for Eqs.(6.23) and (6.29), we add the $(i+2)$-th point control in the expression of M.

For the polynomial distortion model, the projection function Eq.(6.40) does not have a closed-form expression. We apply Subsection 6.4.1 to compute the projection point $\mathbf{p}$ as we did in Section 5.3 and Appendix B.1. For the unified camera model, Eq.(6.40) has a closed-form expression (see Section 4.1.2 and Appendix B.2). Moreover, the correct and efficient computation of the Jacobian is the key to good performance. The derivative calculations are in Appendix B. The difference is that we need to calculate additionally the derivative of $\mathbf{p}$ with respect to neighboring poses $\mathbf{m}_{i-1}, \mathbf{m}_i, \mathbf{m}_{i+1}$ (if we use the approximation Eqs.(6.14) and (6.21)) and with respect to Δ_j and τ. Using the chain rule, the derivative computations can be deduced from the camera motion (Eq.(6.14) or Eq.(6.21)), the time parameter Eq.(6.39) and the partial derivative of $\mathbf{p}$ with respect to multi-camera pose in Appendix B.3. For *Squad* Eq.(6.23) and spline Eq.(6.29), it is, however, difficult to obtain the analytical expression of Jacobian matrix in BA. In practice, we use numerical derivatives for these interpolation methods.

6.4.3 Exact solution

Here we focus on the projection $\mathbf{p}$ by the j-th camera in the i-th keyframe of 3D point $\mathbf{X}$ in the world coordinate system without the approximation in [Klingner+13]: $t_{\mathbf{p}} \neq t_{\tilde{\mathbf{p}}}$ in Eq.(6.36). Let $p_j(\boldsymbol{\theta}_j, \mathbf{m}, \mathbf{X})$ be the projection of $\mathbf{X}$ where $\mathbf{m} = M(t)$ is the parameter of the multi-camera pose in the world coordinates ($k = 3$ for Euler angle or $k = 4$ for quaternion) which is introduced in Section 6.3 and the vector $\boldsymbol{\theta}_j$ is the same as in Section 6.4.2. The acquisition time of $\mathbf{p}$ is

$$t(\Delta_j, \tau, \mathbf{p}) = t_i + \Delta_j + \tau(0 \quad 1)\mathbf{p}. \tag{6.41}$$

and we use notation $M(\mathbf{m}_{i-1}, \mathbf{m}_i, \mathbf{m}_{i+1}, t)$ for the chosen approximation (if we use Eq.(6.14) or Eq.(6.21)). Thus, we have

$$\mathbf{p} = p_j(\boldsymbol{\theta}_j, M(\mathbf{m}_{i-1}, \mathbf{m}_i, \mathbf{m}_{i+1}, t(\Delta_j, \tau, \mathbf{p})), \mathbf{X}). \tag{6.42}$$

Now, we define $\boldsymbol{\theta}$ and g by

$$\boldsymbol{\theta} = (\boldsymbol{\theta}_j, \mathbf{m}_{i-1}, \mathbf{m}_i, \mathbf{m}_{i+1}, \Delta_j, \tau, \mathbf{X}) \text{ and} \tag{6.43}$$

$$g(\mathbf{z}, \boldsymbol{\theta}) = p_j(\boldsymbol{\theta}_j, M(\mathbf{m}_{i-1}, \mathbf{m}_i, \mathbf{m}_{i+1}, t(\Delta_j, \tau, \mathbf{z})), \mathbf{X}) - \mathbf{z}. \tag{6.44}$$

We see that $g(\mathbf{p}, \boldsymbol{\theta}) = \mathbf{0}$ thanks to Eq.(6.42). Then we apply Subsection 6.4.1: find $\mathbf{p}$ by minimizing $\mathbf{z} \mapsto ||g(\mathbf{z}, \boldsymbol{\theta}_0)||^2$ and use Eq.(6.38) for $\mathbf{p}$ derivatives.

If we use the linear trajectory approximation M_1 as in Eq.(6.14), we have a simple expression

$$\frac{\partial g}{\partial \mathbf{z}} = \frac{\partial p_j}{\partial \mathbf{m}} \frac{\partial M}{\partial t} \frac{\partial t(\Delta_j, \tau, \mathbf{z})}{\partial \mathbf{z}} - \mathtt{I}_2 = \tau \frac{\partial p_j}{\partial \mathbf{m}} D_1^i(0 \ 1) - \mathtt{I}_2 \in \mathbb{R}^{2\times 2}. \tag{6.45}$$

We note that the derivative computations without approximation can be deduced from those with approximation (i.e. $\frac{\partial \mathbf{p}}{\partial \boldsymbol{\theta}} = \frac{\partial g}{\partial \boldsymbol{\theta}}(\tilde{\mathbf{p}}, \boldsymbol{\theta}_0)$): replace $\tilde{\mathbf{p}}$ by $\mathbf{p}$ in the derivative by $\boldsymbol{\theta}$ and multiply it on the left side by $-\left(\frac{\partial g}{\partial \mathbf{z}}\right)^{-1}$. We do not implement the exact solution with *Squad* Eq.(6.23) and spline Eq.(6.29) because it is difficult to obtain the analytical expression of the derivative $\frac{\partial g}{\partial \mathbf{z}}$.

Finally, Table 6.1 summarizes our camera motion models and parameterization in Section 6.3 as well as their feasibility for image projection solution and analytical derivative for Jacobian matrix in this section. The Taylor approximations Eqs.(6.14) and (6.21) (in Subsection 6.3.1) are used for the cases: Euler-based ($\mathbb{R}^3$), quaternion ($\mathbb{R}^4$) and translation ($\mathbb{R}^3$). The interpolation *Squad* Eq.(6.23) is used for unit quaternion. Last, Eq.(6.29) is spline on $\mathbb{SE}(3)$, i.e. interpolates both rotation part and translation part. In practice, we use quaternion to parametrize rotation in Eq.(6.29) (we can also use Euler angle or angle-axis representation).

6.5 Comparison of the reduced camera system (RCS) between GS and RS

In this section, we explicit the sparsity of the RCS matrix and compare between GS case (in Chapter 5) and RS (SFA) case in this chapter. This is important for efficient

Motion model	Equation	Param. of $\mathcal{R}$		$\mathbf{t}$(t)	Proj. solution		Jacob
		Euler	Quat		$t_{\mathbf{p}} = t_{\tilde{\mathbf{p}}}$	$t_{\mathbf{p}} \neq t_{\tilde{\mathbf{p}}}$	
Linear approx.	Eq.(6.14)	x	x	x	x	x	ana
Quadratic approx.	Eq.(6.21)	x	x	x	x	x	ana
Squad	Eq.(6.23)		x		x		num
Spline on $\mathbb{SE}(3)$	Eq.(6.29)	(x)	x	x	x		num

TABLE 6.1: Summary of our camera motion models and parametrizations of rotation $\mathcal{R}$, translation $\mathbf{t}$. Proj. solution: approximation solution $t_{\mathbf{p}} = t_{\tilde{\mathbf{p}}}$ and exact solution $t_{\mathbf{p}} \neq t_{\tilde{\mathbf{p}}}$. Jacobian (Jacob): ana - analytical and num - numerical derivatives.

computations.

6.5.1 Notations and global structure of the RCS

First, we define notations. The $m+1$ vectors $\mathbf{m}_i = M(t_i)$ are the parameter of the multi-camera trajectory (Section 6.3), they meet $\mathbf{m}_i \in \mathbb{R}^6$ for the approximations in Section 6.3.1, and we define $\mathbf{M} = (\mathbf{m}_0^\top, \dots, \mathbf{m}_m^\top)^\top \in \mathbb{R}^{6(m+1)}$ (here, we only discuss these approximations and the same methodology is used for the interpolations in Section 6.3.2). Let $\mathbf{m}' \in \mathbb{R}^{m'}$ be the other optimized camera parameters among the intrinsic parameters, the camera poses in multi-camera coordinates, the line delay and the time offsets. Since these other camera parameters are the same at all keyframes, $m' \ll 6(m+1)$. For example, $m+1 = 1000$ and $m' \leq 4 + 4 * 15 = 64$ if the multi-camera has four GoPro cameras: there are one line delay τ, three time offsets $\Delta_1, \Delta_2, \Delta_3$ (Section 6.2) and every camera has parameters $f_x, f_y, u_0, v_0, k_1, \dots, k_5$ for the polynomial distortion model (Section 4.1.1) and 6D pose in multi-camera coordinates. The projection function of $\mathbf{X}^l$ (in world coordinates) in the i-th keyframe is concisely written $\varphi(\mathbf{m}_{i-1}, \mathbf{m}_i, \mathbf{m}_{i+1}, \mathbf{m}', \mathbf{X}^l)$ for both approximations in Section 6.3.1 (omit $\mathbf{m}_{i-1}$ if $i = 0$ and omit $\mathbf{m}_{i+1}$ if $i = m$).

Let matrix $\mathtt{A}$ be the Jacobian matrix with respect to all extrinsic, intrinsic, line delay and time offsets, $\mathtt{B}$ be the Jacobian matrix with respect to all multi-camera keyframe poses, $\mathtt{C}$ be the Jacobian matrix with respect to all 3D points and $\boldsymbol{\epsilon}$ be the residual vector. And we use the notations (they are almost the same in Section 5.4): $\mathtt{Q} = \mathtt{A}^\top\mathtt{A}$, $\mathtt{U} = \mathtt{B}^\top\mathtt{B}$, $\mathtt{V} = \mathtt{C}^\top\mathtt{C}$, $\mathtt{E} = \mathtt{A}^\top\mathtt{B}$, $\mathtt{F} = \mathtt{A}^\top\mathtt{C}$, $\mathtt{W} = \mathtt{B}^\top\mathtt{C}$. The Jacobian matrix is partitioned as $\mathtt{J} = [\mathtt{A}|\mathtt{B}|\mathtt{C}]$. The approximated Hessian $\mathtt{H}$ of the cost function minimized by BA is expressed with block matrices

$$\mathtt{H} = \mathtt{J}^\top\mathtt{J} = \left[\begin{array}{cc|c} \mathtt{Q} & \mathtt{E} & \mathtt{F} \\ \mathtt{E}^\top & \mathtt{U} & \mathtt{W} \\ \hline \mathtt{F}^\top & \mathtt{W}^\top & \mathtt{V} \end{array}\right] \tag{6.46}$$

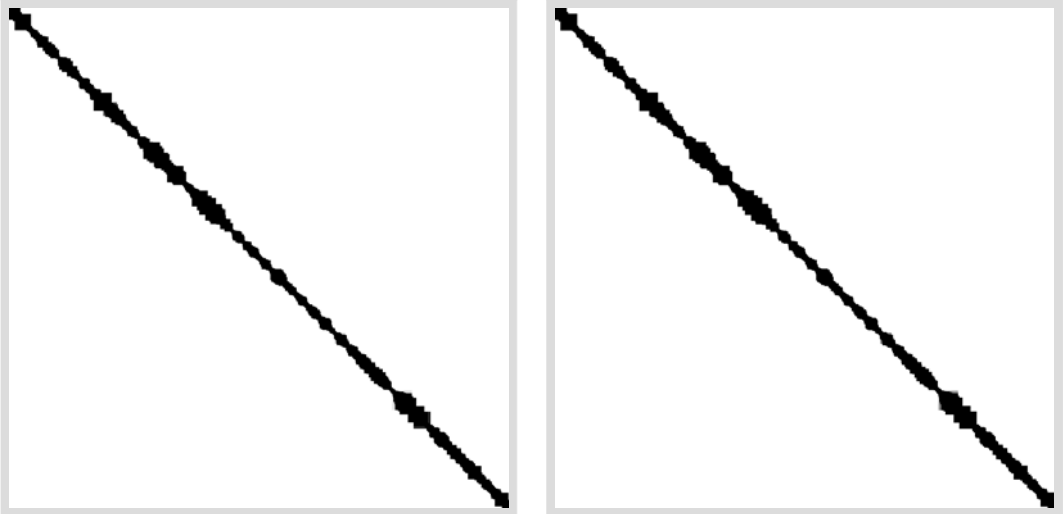
As mentioned in Section 5.4, the RCS is

$$\mathtt{S} = \begin{bmatrix} \mathtt{Q} & \mathtt{E} \\ \mathtt{E}^\top & \mathtt{U} \end{bmatrix} - \begin{bmatrix} \mathtt{F} \\ \mathtt{W} \end{bmatrix} \mathtt{V}^{-1} \begin{bmatrix} \mathtt{F} \\ \mathtt{W} \end{bmatrix}^\top = \begin{bmatrix} \mathtt{Z}'' & \mathtt{Z}' \\ (\mathtt{Z}')^\top & \mathtt{Z} \end{bmatrix}. \tag{6.47}$$

We have $\mathtt{Z} = \mathtt{U} - \mathtt{W}\mathtt{V}^{-1}\mathtt{W}^\top \in \mathbb{R}^{6(m+1)\times 6(m+1)}$, $\mathtt{Z}' \in \mathbb{R}^{6(m+1)\times m'}$ and $\mathtt{Z}'' \in \mathbb{R}^{m'\times m'}$. Since $m' \ll 6(m+1)$, $\mathtt{Z}$ is the preponderant block in the RCS and we only focus on the $\mathtt{Z}$ sparsity.

6.5.2 Sparsity of Z

Here we present Z by a shape included in $\mathbb{Z}^2$, i.e. a set of pixels in an image such that every pixel corresponds to a non-zero 6×6-block of Z. Then we show in Appendix H that the shape of our Z (which involves SFA and RS using projection function $\varphi(\mathbf{m}_{i-1}, \mathbf{m}_i, \mathbf{m}_{i+1}, \mathbf{m}', \mathbf{X}^l)$) is included in a dilation of the shape of the standard Z (which involves FA and GS using the projection function $\varphi(\mathbf{m}_i, \mathbf{m}', \mathbf{X}^l)$ in Chapter 5) by $\{-1, 0, +1\}^2$. We remind that this dilation is an operation morphology that expands a shape by one pixel in both dimensions and both directions. Thus our RCS is slightly less sparse that the standard RCS (in practice it is very similar according to the example in Figure 6.7).



(a) Standard (GS, FA) case (b) (RS, SFA) case

FIGURE 6.7: Shape of the standard (GS, FA) and (RS, SFA) Z for video sequence BC1 with 2047 keyframes and same inliers without loop closure. Each black pixel corresponds to a non-zero 6×6-block of Z.

In the case where the loops are not closed in the video and the track length is bounded by l, the standard Z (GS case) is a 6×6-block-wise band matrix with bandwidth l (Sec. A6.7.1 in [Hartley+04]) and our Z in (RS, SFA) case is a 6×6-block-wise band matrix with bandwidth $l + 1$.

6.6 Experiments

6.6.1 Main notations

In this section, our BA is named by a combination of several notations that describes the estimated parameters:

- C (central approximation) estimates all rotations $\mathtt{R}_{\mathrm{C}}^j$ and fixes all translations $\mathbf{t}_{\mathrm{C}}^j = \mathbf{0}$,
- NC (non-central) estimates all $(\mathtt{R}_{\mathrm{C}}^j, \mathbf{t}_{\mathrm{C}}^j)$,

- INT (intrinsic) estimates all intrinsic parameters: $(f_x, f_y, u_0, v_0, k_1, \ldots, k_5)$ or $(f_x, f_y, u_0, v_0, \xi)$ depending on the camera model; every camera has its own parameters,
- FA (frame accurate) fixes all time offsets $\Delta_j = 0$,
- SFA (sub-frame accurate) estimates all Δ_j,
- GS (global shutter) fixes the line delay $\tau = 0$,
- RS (rolling shutter) estimates τ.

Thus GS.NC.SFA.INT (or gs.nc.sfa.int) is a BA that fixes $\tau = 0$ and estimates simultaneously all $\Delta_j, \mathtt{R}_{\mathrm{C}}^j, \mathbf{t}_{\mathrm{C}}^j$, intrinsic parameters, keyframe poses $\mathbf{m}_i$ and 3D points. The threshold for inlier selection is set to 4 pixels in all videos. Every BA has three inlier updates, each one is followed by Levenberg-Marquardt minimization for these inliers. A succession of two BAs is possible, e.g. gs.c.fa.int+rs.c.sfa.

We use shortened notations:

- #2D = number of 2D inliers,
- GT = ground truth,
- f = FpS (in Table 3.1),
- method gs.sfa is the SFA refinement in Subsection 4.2.3 (for comparison),
- $v_{\max}$ is the number of lines of a monocular image (in Table 6.2 below).

Table 6.2 summarizes our datasets (both cameras and videos) and complements to complete Table 3.1. We would like to highlight the ground truth. BC2 has ground truth: $f\Delta_1 = 0.25$, $f\Delta_2 = 0.5$, $f\Delta_3 = 0.75$ and $\tau = 9.12\mu$s (reminder: if $f\Delta_j = 1$, Δ_j is the time between two consecutive frames). CC has also complete ground truth. The other videos (BC1, WT, FH, WU) have incomplete ground truth (a strobe always provides τ, see Appendix G).

Name (short name)	Camera	$v_{\max}$	$\tau(\mu s)$	$f\Delta_j$	kfr	#Tracks	$\|\|\beta_i\|\|_\infty$
BikeCity1 (BC1)	4*Gopro 3	960	9.10	?	2047	343k	0.223
WalkTown (WT)	4*Gopro 3	960	9.10	?	1363	240k	0.268
FlyHill (FH)	4*Gopro 3	1440	11.3	?	627	432k	0.494
BikeCity2 (BC2)	4*Gopro 3	960	9.12	i/4	225	51k	0.074
CarCity (CC)	Ladybug 2	768	0	0	891	282k	0.068
WalkUniv (WU)	Theta S	960	-32.1	0	1287	154k	0.129

TABLE 6.2: Datasets: number of keyframes kfr, maximum of angles $|\beta_i|$ of our Euler angles parametrization, number of lines $v_{\max}$ of a monocular image, line delay τ (ground truth), time offset $f\Delta_j$ (ground truth), number of tracks #Tracks (3D points).

Error $e(\Delta)$ is the sum of absolute errors of all $f\Delta_j$. Error $e(\tau)$ is the relative error of τ. We also would like to estimate the error of a multi-camera calibration by a single number d defined in Subsection 5.6.1. We remind that d is based on the angle between rays of the two calibrations (the estimated one and the GT one) that have the same

pixel. In some cases, results are represented by the symbols: "na" (not available) for missing data, "nan" (not a number) for impossible values (e.g. dividing by zero), "?" for unknown values (e.g., we do not have ground truth for this estimate).

Note that Section 6.3 presents two approximations (for Euler-based as well as unit quaternions) and two interpolations (for unit quaternions parametrization and for Lie group $\mathbb{SE}(3)$) for multi-camera trajectories. Section 6.4 describes the approximate ($t_{\mathbf{p}} = t_{\tilde{\mathbf{p}}}$) and exact ($t_{\mathbf{p}} \neq t_{\tilde{\mathbf{p}}}$) calculation for image projection with RS and SFA synchronization (see Table 6.1). The experiments of this chapter are organized as follows:

- In the four first Subsections 6.6.2-6.6.5, we show the performance of our BA using the simplest method: the linear approximation Eq.(6.14) for camera motion using Euler angle and the approximation $t_{\mathbf{p}} = t_{\tilde{\mathbf{p}}}$.
- The results for all videos using quadratic approximation or spherical interpolations of $M(t)$ or using Subsection 6.4.3 (i.e. without approximation $t_{\mathbf{p}} = t_{\tilde{\mathbf{p}}}$) are compared in Subsection 6.6.6.
- An experiment in the case of Euler angle that is close to a singularity is presented in the Subsection 6.6.7.
- The last subsection shows the experiment in the case of fast camera motion.

6.6.2 Rolling shutter and subframe-accurate synchronization

First, Subsection 6.6.2.1 provides all estimation errors of several BAs for videos BC2 and CC that have complete ground truth. Second, Subsection 6.6.2.2 provides SFA time offsets, line delay and top view of reconstruction for every video using RS.C.SFA.INT.

6.6.2.1 Accuracies

Table 6.3 provides errors $e(\Delta)$, $e(\tau)$ and d for several BAs estimating both SFA time offsets Δ_j and line delay τ. We compare GS.C.FA.INT+RS.C.SFA and RS.C.SFA.INT, i.e. we compare separate and simultaneous estimations of INT and RS.SFA parameters. We also compare these central BAs and their non-central versions, and the SFA synchronization without BA in Subsection 4.2.3.

First, we experiment on the only RS sequence that has complete ground truth - BC2. We see that the simultaneous estimation of INT and RS.SFA has quite smaller $e(\tau)$ and smaller d than separate estimations (both C and NC BAs). However, the separate case has a twice smaller $e(\Delta)$ than the simultaneous case, which in turn is more than twice smaller than that of the SFA refinement without BA. We remind that $e(\Delta)$ cumulates absolute errors of SFA synchronization: a value of 0.1 (for simultaneous case) means that the mean SFA sync. error of n cameras is only $(0.1/(n-1))$. The NC BAs also greatly reduce d.

Second, we experiment on CC which is the only real sequence with complete ground truth. Since it is GS, relative error $e(\tau)$ is not a number and we replace it by $v_{\max} f \tau$ (the smaller absolute value, the better result). We see that all BAs provide small $|v_{\max} f \tau|$ compared to that of consumer RS cameras: we obtain values in $[0.002, 0.009]$, which are

Method applied to BC2	$e(\Delta)$	$e(\tau)$	d
gs.c.fa.int+rs.c.sfa	0.057	14.6%	1.970
rs.c.sfa.int	0.097	2.7%	1.476
gs.nc.fa.int+rs.nc.sfa	0.051	12.2%	1.312
rs.nc.sfa.int	0.111	3.7%	0.366
gs.sfa (in Subsec.4.2.3)	0.215	na	na
Method applied to CC	$e(\Delta)$	$v_{\max} f\tau$	d
gs.c.fa.int+rs.c.sfa	0.052	-0.0052	1.176
rs.c.sfa.int	0.055	-0.0069	1.167
gs.nc.fa.int+rs.nc.sfa	0.034	-0.0020	1.322
rs.nc.sfa.int	0.039	0.0086	1.313
gs.sfa (in Subsec.4.2.3)	6e-3	na	na

TABLE 6.3: Accuracies of rs.X.sfa.Y(int) for BC2 and for CC.

small in comparison to typical values $[0.8 - 0.9]$ of consumer RS cameras (see Table 6.2 or the next section). Furthermore, all $e(\Delta)$ are smaller that 0.055 for five cameras; d increases and $e(\Delta)$ decreases by NC BAs. The SFA refinement without BA (in Subsection 4.2.3) provides the smallest $e(\Delta)$ in this GS dataset.

6.6.2.2 Time offsets, line delays and reconstruction

Table 6.4 shows time offsets $f\Delta_j$, normalized line delay $v_{\max} f\tau$ and error $e(\tau)$ for all videos by applying RS.C.SFA.INT. We see that error $e(\tau)$ is less than 7.2% except for WT. In the WT case, τ is over-estimated ($v_{\max} f\tau$ is even greater than its theoretical maximum value 1) and has large error $e(\tau)$ equal to 16%. In contrast to this, $e(\tau)$ in the BC1 case looks lucky. In fact, the τ value in a single experiment should be moderated since it depends on the keyframe choice (this will be experimented in Subsection 6.6.3). Note that a negative value of τ (for WU) simply means that the time of the v-th line increases when v decreases.

Name	$f\Delta_1$	$f\Delta_2$	$f\Delta_3$	$v_{\max} f\tau$	GT	$e(\tau)$
BC1	-0.334	-0.153	0.132	0.8755	0.8736	0.2%
WT	-0.583	-0.320	-0.795	1.0136	0.8736	16.0%
FH	0.287	0.203	-0.326	0.8372	0.7810	7.2%
BC2	0.246	0.546	0.797	0.8989	0.8755	2.7%
CC	-0.017	-0.013	-0.006	-0.0069	0	nan
WU	0.001	na	na	-0.8882	0.9244	3.9%

TABLE 6.4: SFA time offsets and line delay accuracy for all datasets using rs.c.sfa.int. Here GT is the ground truth of $v_{\max} f\tau$.

Figures 6.8 and 6.9 show a top view of the RS.C.SFA.INT reconstructions (both keyframes locations and 3D point cloud). In the WU case, we observe a non-negligible drift since the beginning and end of trajectory should be the same (the drift is less noticeable in the other examples). There are several reasons: we do not enforce loop closure, the incremental multi-camera SfM by local BA [Mouragnon+09] is done using an intermediate calibration computed from only 2k first frames, and the final BA (RS.C.SFA.INT) does not remove the drift. We redo the incremental SfM using the final

multi-camera calibration (computed from the whole sequence by RS.C.SFA.INT) and see that an important part of drift is removed. This suggests that the final multi-camera calibration is better than the intermediate one.

6.6.3 Stability of synchronization and rolling shutter with respect to keyframe sampling

Now we experiment the stability of our results (SFA synchronization, line delay and calibration) with respect to moderated changes of keyframe sampling. The keyframe sampling is tuned by a single threshold N_3, which is a lower bound for the number of matches between three consecutive keyframes (more details in Subsection 2.4.1). For every value $N_3 \in \{400, 425, 450, 475, 500\}$, we apply multi-camera SfM based on keyframe sampling followed by RS.C.SFA.INT and then discuss the results. The initial multi-camera calibration and FA synchronization are the same for all N_3 and are computed from the video beginning as in the other experiments.

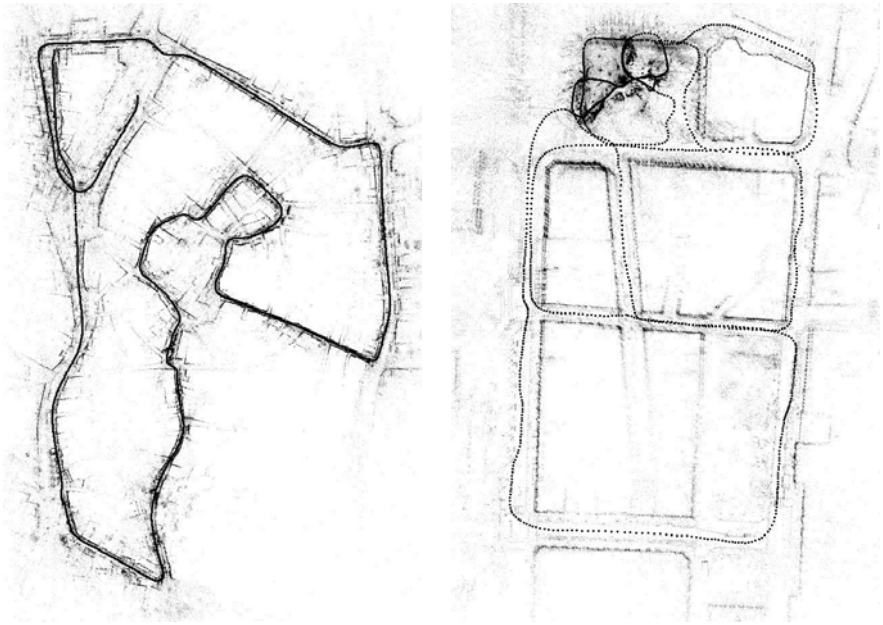
	rs.c.sfa.int			rs.c.sfa.int+rs.c.sfa.int.h		
N_3	$e(\Delta)$	$e(\tau)$	d	$e(\Delta)$	$e(\tau)$	d
400	0.089	1.9%	1.466	0.091	2.4%	1.458
425	0.131	3.0%	1.570	0.123	0.8%	1.563
450	0.097	2.7%	1.476	0.110	2.8%	1.541
475	0.191	7.4%	1.867	0.136	5.5%	1.710
500	0.199	2.7%	1.664	0.101	0.4%	1.499
mean	0.141	3.5%	1.609	0.112	2.4%	1.554
max/min	0.046	4.0	1.274	0.016	12.3	1.17

TABLE 6.5: Accuracies stability with respect to keyframe sampling for BC2 using rs.c.sfa.int (left) and rs.c.sfa.int+rs.c.sfa.int.h (right) with h = 70%. There are 206 keyframes if $N_3 = 400$ and 248 keyframes if $N_3 = 500$.

The left of Table 6.5 shows estimation errors $e(\Delta)$, $e(\tau)$ and d. There are 206 keyframes if $N_3 = 400$ and 248 keyframes if $N_3 = 500$. The variations of errors are important: from single to double for $e(\Delta)$, from single to quadruple for $e(\tau)$, and about 30% for d. We provide an explanation in Subsection 6.6.3.1 and a correction of the results in Subsection 6.6.3.2. Table 6.6 shows times offsets $f\Delta_j$ and error $e(\tau)$ for the longest sequence BC1. The variations of $e(\tau)$ are also important; the variations of $f\Delta_j$ are less than 0.032.

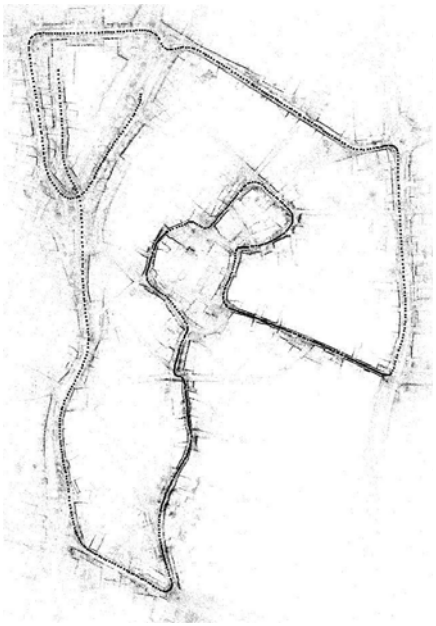
N_3	kfr	$f\Delta_1$	$f\Delta_2$	$f\Delta_3$	$v_{\max} f\tau$	$e(\tau)$
400	1813	-0.341	-0.171	0.131	0.8920	2.1%
425	1929	-0.337	-0.155	0.131	0.8882	1.6%
450	2047	-0.334	-0.153	0.132	0.8755	0.2%
475	2166	-0.366	-0.156	0.134	0.9399	7.5%
500	2256	-0.337	-0.141	0.130	0.8786	0.5%
mean	2042	-0.343	-0.155	0.132	0.8948	2.4%
max-min	443	0.032	0.030	0.004	0.0644	2.1%

TABLE 6.6: Stabilities of time offsets and line delay with respect to keyframe sampling for BC1 using rs.c.sfa.int. The number of keyframe is kfr.

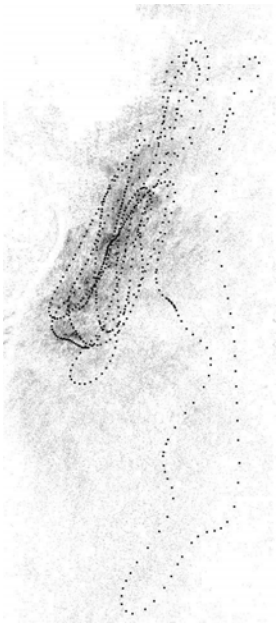


(a) BC1

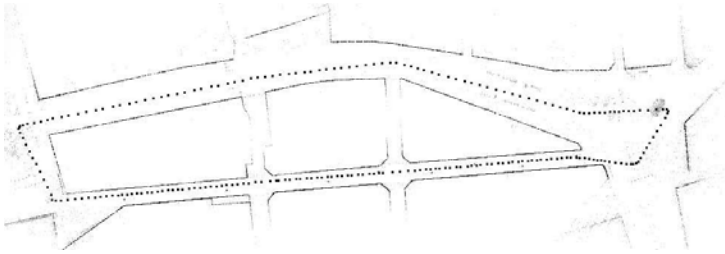
(b) WT



(c) CC

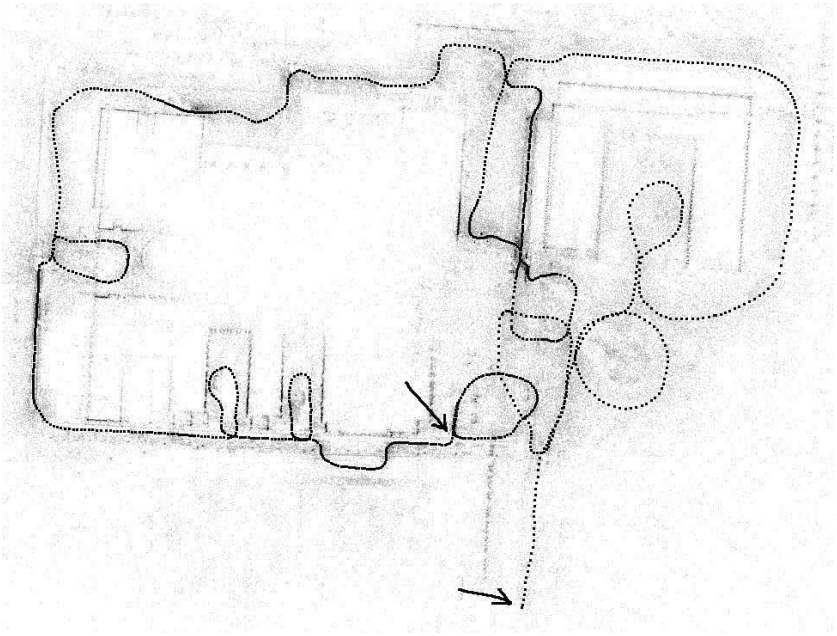


(d) FH

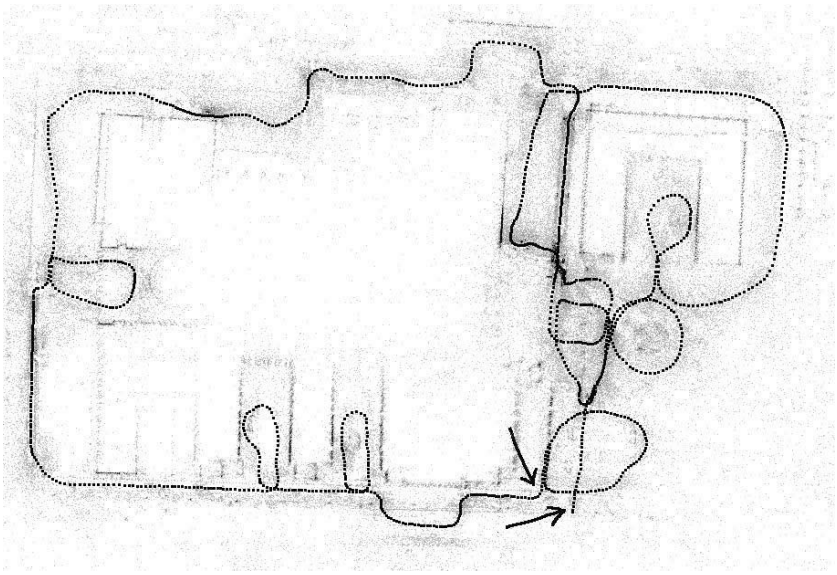


(e) BC2

FIGURE 6.8: Top views of RS.C.SFA.INT reconstructions of BC1, WT, FH, BC2 and CC without loop closure. The input videos of BC1, WT and FH are taken by four GoPro cameras mounted on a helmet. The FH trajectory has a lot of sharp "S" turns.



(a) RS.C.SFA.INT



(b) Incremental SfM using the calibration estimated by RS.C.SFA.INT

FIGURE 6.9: Top views of RS.C.SFA.INT reconstructions of WU without loop closure. The input video is taken by the Ricoh Theta S mounted on a helmet. The drift is between the two arrows.

6.6.3.1 Analysis

We remind that the reprojection errors in the i-th keyframe are computed using an approximation of multi-camera trajectory $M(t)$ where $t \approx t_i$: $M(t)$ is a linear combination of $\mathbf{m}_{i-1}$, $\mathbf{m}_i$ and $\mathbf{m}_{i+1}$ and M is linear (Eq.(6.14)) in time $t - t_i$. The better this approximation, the smaller the reprojection error in the i-th keyframe.

At first glance, the estimation errors decrease if N_3 increases: if N_3 increases, the keyframe density increases, thus the accuracy of these approximations is better

(the remainders $\mathcal{O}(\delta)$ and $\mathcal{O}(\delta^2)$ in Subsection 6.3.1 decrease), the reprojection errors decrease and last the estimation errors decreases. However, the errors in Table 6.5 are not decreasing series but look noisy.

Here is a second explanation. The true value of $M(t)$ near t_i can be different to its approximated value for some i, e.g. if the true speed vector $\dot{M}(t_i)$ is different to speed vector D_1 computed using the multi-camera poses of neighboring keyframes $i-1$ and $i+1$ in Eq.(6.13). Then a keyframe with bad approximation has high reprojection errors and acts as an outlier perturbing the BA. The estimation errors in the left of Table 6.5 depend on the set of this kind of outliers, which in turn depends on N_3.

6.6.3.2 Correction

The idea is simple: if the i-th keyframe has high reprojection errors, we redefine its approximation by $M(t) = \mathbf{m}_i + (t - t_i)\mathbf{d}_i$ if $t \approx t_i$ thanks to new velocity parameter $\mathbf{d}_i \in \mathbb{R}^6$ that is estimated by BA like $\mathbf{m}_i$. Then the camera motion is not constrained by keyframes $i-1$ and $i+1$ if $t \approx t_i$, and we expect that the resulting reprojection errors of the i-th keyframe decrease such that the i-th keyframe does not act as an outlier of the BA.

In practice, we start from a current estimation obtained by RS.C.SFA.INT and introduce a user defined percentage h. Let c_h be the h-fractile over all reprojection errors. For every keyframe, we compute the RMS of its own reprojection errors. If this RMS is greater than c_h, the keyframe has an additional velocity parameter $\mathbf{d}_i$ as above (otherwise it does not have). We name RS.C.SFA.INT.h the new BA obtained by modifying RS.C.SFA.INT like this.

Table 6.5 provides estimation errors $e(\Delta)$, $e(\tau)$, d of both RS.C.SFA.INT and RS.C.SFA.INT+RS.C.SFA.INT.h using $h = 70\%$. Thanks to the correction, all errors are improved in the following sense: both mean and maximum of every error are reduced, the variations of $e(\Delta)$ and d are damped (the variations of $e(\tau)$ expressed using ratio max/min are not damped due to a small error 0.4% for $N_3 = 500$).

Lastly, Table 6.7 shows time offsets $f\Delta_j$, normalized line delay $v_{\max} f\tau$ and error $e(\tau)$ for all videos by applying RS.C.SFA.INT+RS.C.SFA.INT.h. We see that all errors $e(\tau)$ have same magnitude order (in interval [2.8, 8]%). This contrasts to Table 6.4 using $N_3 = 450$, where $e(\tau)$ is small for BC1 and large for WT.

Name	$\#\mathbf{d}_i$	$f\Delta_1$	$f\Delta_2$	$f\Delta_3$	$v_{\max} f\tau$	GT	$e(\tau)$
BC1	141	-0.349	-0.152	0.112	0.9177	0.8736	5.0%
WT	126	-0.541	-0.322	-0.789	0.9139	0.8736	4.6%
FH	132	0.284	0.208	-0.329	0.8435	0.7810	8.0%
BC2	15	0.261	0.548	0.801	0.9001	0.8755	2.8%
CC	3	-0.004	-0.015	-0.011	-0.0009	0	nan
WU	131	-0.001	na	na	-0.8772	0.9244	5.1%

TABLE 6.7: SFA time offsets and line delay accuracy for all datasets using rs.c.sfa.int+rs.c.sfa.int.h with $h = 70\%$ (to be compared with Table 6.4). Here GT is the ground truth of $v_{\max} f\tau$. The number of keyframes with additional velocity parameter is $\#\mathbf{d}_i$.

6.6.4 Robustness with respect to FA synchronization

We would like to know whether an error in the FA synchronization can be corrected by SFA synchronization. Such an error can have two reasons: the IAV variations are insufficient in the video beginning for FA synchronization, or a consumer camera skips frame(s) for any technical reasons. We simulate such an error by skipping x frames of camera 1 (reminder: camera 0 is the first one), then we use multi-camera SfM followed by RS.C.SFA.INT and compare the results for $x \in \{0, 1, 2, 3\}$.

x	$f\Delta_1$	$f\Delta_2$	$f\Delta_3$	$v_{\max} f\tau$	$e(\Delta)$	$e(\tau)$	d
0	0.246	0.546	0.797	0.8989	0.097	2.7%	1.476
1	1.342	0.505	0.769	0.8666	0.117	1.0%	1.743
2	2.026	0.489	0.760	0.7834	0.245	10.5%	1.298
3	2.472	0.532	0.781	0.7848	0.841	10.4%	3.244

TABLE 6.8: Accuracies of rs.c.sfa.int applied to BC2 if we shift x additional frames(s) of camera 1. The ideal results meets $f\Delta_1 = x + 0.25$.

Table 6.8 shows errors $e(\Delta)$, $e(\tau)$, d and time offsets $f\Delta_j$ estimated for BC2. We see that $e(\Delta)$ and d increases moderately if $x = 1$, $e(\Delta)$ is multiplied by 2.5 and $e(\tau)$ by 3.9 if $x = 2$, and all errors increase a lot if x = 3.

6.6.5 Variations of line delay and SFA time offsets in a long sequence

Here we examine the variations of τ and Δ_j in a long sequence. We split the GS.C.FA.INT reconstruction of BC1 into six segments of 300 keyframes (segments 0-299, 300-599, etc) and independently apply RS.C.SFA.INT to every segment. Table 6.9 shows the results.

The variations of $f\Delta_j$ are moderated (less than 0.2) and the $f\Delta_j$s globally increase over time, i.e. when s increases. At first glance, we could expect that we can detect a frame skipped by a camera (if any) by an increase/decrease of 1 as in Subsection 6.6.4. However, this is not the case. Since all cameras are in the same manufacturing series and have the same setting, all cameras skip similar numbers of frames (if any) in a segment, which in turn would imply that we do not observe large time offset perturbations. Furthermore, we could interpret the slow increase of the $f\Delta_j$s as follows: the FpS of camera 0 is slightly lower than those of the other cameras.

The variations of τ are important (especially in the first half part of BC1) and τ globally decreases over time. The variations of τ are reduced if we take a larger video segment size, e.g. we divide by two the $v_{\max} f\tau$ range (max-min in Table 6.9) with 500 keyframes per segment.

$D = 300$					
s	$f\Delta_1$	$f\Delta_2$	$f\Delta_3$	$v_{\max} f\tau$	$e(\tau)$
0	-0.443	-0.180	0.014	1.0570	20.1%
1	-0.417	-0.152	0.014	0.9347	6.9%
2	-0.306	-0.073	0.028	0.7678	12.2%
3	-0.308	-0.163	0.115	0.9225	5.6%
4	-0.271	-0.157	0.178	0.9056	3.7%
5	-0.299	-0.105	0.199	0.8394	3.9%
mean	-0.341	-0.138	0.091	0.9045	8.7%
max-min	0.172	0.107	0.185	0.2892	16.4%
$D = 500$					
s	$f\Delta_1$	$f\Delta_2$	$f\Delta_3$	$v_{\max} f\tau$	$e(\tau)$
0	-0.432	-0.172	0.009	1.0123	15.8%
1	-0.337	-0.138	0.093	0.8773	0.4%
2	-0.300	-0.172	0.132	0.9161	4.8%
3	-0.297	-0.116	0.209	0.8708	0.3%
mean	-0.342	-0.150	0.111	0.9191	5.4%
max-min	0.135	0.056	0.200	0.1415	15.5%

TABLE 6.9: Stabilities of time offsets and line delay over time in long sequence BC1 using rs.c.sfa.int. The s-th video segment is taken between keyframes Ds and $Ds+D-1$.

6.6.6 Comparing parameterizations of the multi-camera motion

All previous experiments are done using the linear approximation of $M(t)$ (Eq.(6.14)) for Euler-based angle and using the approximation $t_{\mathbf{p}} = t_{\tilde{\mathbf{p}}}$ in Eq.(6.39). We remind that all our motion models are listed in Table 6.1. Table 6.10 shows the results for all videos using quadratic approximation of $M(t)$ (Eq.(6.21)) or using Subsection 6.4.3 (i.e. without approximation $t_{\mathbf{p}} = t_{\tilde{\mathbf{p}}}$ in Eq.(6.39)) or using quaternion parametrization. We do not observe significant improvements of $e(\tau)$ in comparison to those in Table 6.4 except for FH using Eq.(6.21). By recomputing errors $e(\Delta)$ and d for BC2 and CC using these three changes (for Euler angle parametrization, linear approx. $\rightarrow$ quadratic approx. or $t_{\mathbf{p}} = t_{\tilde{\mathbf{p}}} \rightarrow t_{\mathbf{p}} \neq t_{\tilde{\mathbf{p}}}$; or Euler-based $\rightarrow$ quaternion), we obtain very similar results as in Table 6.3: $e(\Delta)$ difference is less than 0.008 and d difference is less than 0.042.

Table 6.11 provides the results for Gopro multi-camera using spherical interpolations for camera motion with the time approximation $t_{\mathbf{p}} = t_{\tilde{\mathbf{p}}}$. For synthetic dataset BC2, we obtain slightly improvement in terms of $e(\tau)$ and $e(\Delta)$ using *Squad*, but no improvement using spline on $\mathbb{SE}(3)$. There is no significant change in terms of d. For other videos, we keep an eye on $e(\tau)$. The spherical interpolations provide similar result for WT. But the relative error of τ is worse for BC1 and clearly better for FH than that in Table 6.4 and 6.10. There are several reasons. FH has the fastest image motion among our videos. Due to sharp "S" turns, spherical interpolations are more precise than Taylor approximations for this case thanks to its complexity. This also confirms that the linear Taylor approximation Eq.(6.14) is efficient for camera motion model in many usual cases (at least in the case of bike riding or walking).

All the experiments in this thesis are performed on a machine with an Intel® Core™ i7-3770 CPU @ 3.40GHz with 16 GB of RAM. We compare the execution

Name	$f\Delta_1$	$f\Delta_2$	$f\Delta_3$	$v_{\max} f\tau$	GT	$e(\tau)$	$e(\Delta)$	d
rs.c.sfa.int (Euler-based, linear approx, $t_{\mathbf{p}} = t_{\tilde{\mathbf{p}}}$) (ref. Tables 6.4 and 6.3)								
BC1	-0.334	-0.153	0.132	0.8755	0.8736	0.2%	?	?
WT	-0.583	-0.320	-0.795	1.0136	0.8736	16.0%	?	?
FH	0.287	0.203	-0.326	0.8372	0.7810	7.2%	?	?
BC2	0.246	0.546	0.797	0.8989	0.8755	2.7%	0.097	1.476
CC	-0.017	-0.013	-0.006	-0.0069	0	nan	0.055	1.167
WU	0.001	na	na	-0.8882	0.9244	3.9%	0.001	?
rs.c.sfa.int (Euler-based, quadratic approx, $t_{\mathbf{p}} = t_{\tilde{\mathbf{p}}}$)								
BC1	-0.337	-0.157	0.127	0.8624	0.8736	1.3%	?	?
WT	-0.580	-0.323	-0.791	0.9974	0.8736	14.2%	?	?
FH	0.284	0.201	-0.326	0.8252	0.7810	5.7%	?	?
BC2	0.249	0.551	0.798	0.9020	0.8755	3.0%	0.100	1.476
CC	-0.017	-0.013	-0.005	-0.0082	0	nan	0.054	1.175
WU	-2e-4	na	na	-0.8896	-0.9244	3.8%	-2e-4	?
rs.c.sfa.int (Euler-based, linear approx, $t_{\mathbf{p}} \neq t_{\tilde{\mathbf{p}}}$)								
BC1	-0.334	-0.153	0.131	0.8758	0.8736	0.3%	?	?
WT	-0.581	-0.320	-0.793	1.0061	0.8736	15.2%	?	?
FH	0.287	0.203	-0.326	0.8381	0.7810	7.3%	?	?
BC2	0.245	0.547	0.796	0.9028	0.8755	3.1%	0.098	1.475
CC	-0.017	-0.014	-0.005	-0.0096	0	nan	0.055	1.177
WU	0.001	na	na	-0.8689	-0.9244	6.0%	0.001	?
rs.c.sfa.int (Quaternion, linear approx, $t_{\mathbf{p}} = t_{\tilde{\mathbf{p}}}$)								
BC1	-0.336	-0.154	0.131	0.8800	0.8736	0.7%	?	?
WT	-0.584	-0.322	-0.799	1.0158	0.8736	16.3%	?	?
FH	0.290	0.206	-0.333	0.8485	0.7810	8.6%	?	?
BC2	0.248	0.550	0.803	0.9038	0.8755	3.2%	0.105	1.475
CC	-0.021	-0.009	-0.011	0.0063	0	nan	0.059	1.125
WU	0.002	na	na	-0.8970	-0.9244	3.7%	0.002	?

TABLE 6.10: SFA time offsets and line delay accuracy for all datasets using rs.c.sfa.int with/without approximation $t_{\mathbf{p}} = t_{\tilde{\mathbf{p}}}$, using linear or quadratic approximation. "?" means that GT is unknown for this estimate.

time (in seconds) using RS.C.SFA.INT method with different configurations: rotation parametrization, time approximation as well as camera motion model. Table 6.12 provides comparison on the computation time for BC2 video (225 keyframes, 50k 3D points, 210k 2D points). For all cases, we only run one inlier update followed by LM with 100 iterations for comparison (Note that our normal BA has 3 inlier updates, each one is followed by LM algorithm for these inliers with $k_{\max} = 200$ iterations). As mentioned in Section 6.4 and Table 6.1, we do perform analytical Jacobian for approximations (Eq.(6.14) and Eq.(6.21)) and numerical Jacobian for spherical interpolations (*Squad* Eq.(6.23) and spline on $\mathbb{SE}(3)$ Eq.(6.29)). We can see in Table 6.12 that analytical Jacobian reduces significantly computation time. In practice, the correct and efficient computation of the Jacobian matrix is the key to good performance. Moreover, approximations (Eq.(6.14) and Eq.(6.21)) are functions of 3 neighborhood poses while spherical interpolations (Eq.(6.23) and $\mathbb{SE}(3)$ Eq.(6.29)) yield an interpolation from 4 neighborhood poses. This is the reason why Jacobian density increases for the latter in comparison to the former. At least for this reason, the computation time is longer.

Name	$f\Delta_1$	$f\Delta_2$	$f\Delta_3$	$v_{\max} f\tau$	GT	$e(\tau)$	$e(\Delta)$	d
rs.c.sfa.int (Quaternion, Squad, $t_{\mathbf{p}} = t_{\tilde{\mathbf{p}}}$)								
BC1	-0.373	-0.177	0.117	0.9267	0.8736	6.1%	?	?
WT	-0.584	-0.319	-0.794	1.0048	0.8736	15.0%	?	?
FH	0.273	0.196	-0.311	0.7955	0.7810	1.9%	?	?
BC2	0.252	0.547	0.777	0.8951	0.8755	2.2%	0.075	1.486
rs.c.sfa.int (Quaternion, $\mathbb{SE}(3)$, $t_{\mathbf{p}} = t_{\tilde{\mathbf{p}}}$)								
BC1	-0.372	-0.170	0.135	0.9444	0.8736	8.1%	?	?
WT	-0.588	-0.320	-0.803	1.0194	0.8736	16.7%	?	?
FH	0.272	0.193	-0.306	0.7979	0.7810	2.2%	?	?
BC2	0.261	0.550	0.789	0.9094	0.8755	3.9%	0.100	1.478

TABLE 6.11: SFA time offsets and line delay accuracy for Gopro datasets using rs.c.sfa.int with time approximation using spherical interpolations of quaternion (to be compared with Table 6.4 and 6.10). "?" means that GT is unknown for this estimate.

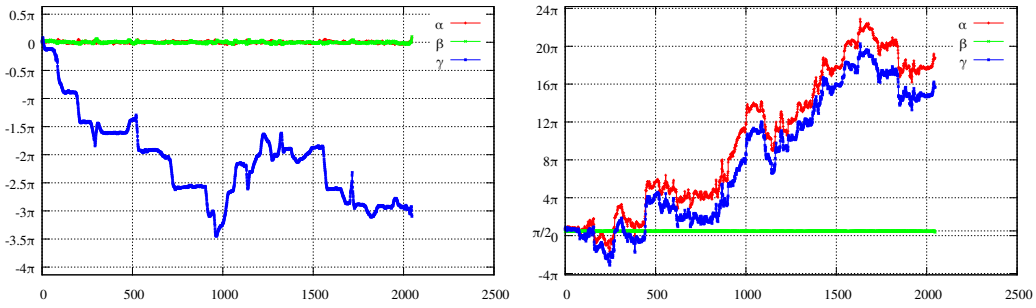
Param	$t_{\mathbf{p}} = t_{\tilde{\mathbf{p}}}$	Motion model	Jacob_eval	Linear solver	Total
Euler	x	linear approx.	20.59	27.61	75.75
Euler	x	quadratic approx.	20.47	27.55	75.42
Euler		linear approx.	35.96	27.42	106.11
Euler		quadratic approx.	35.67	27.23	105.59
Quat	x	linear approx.	20.82	27.53	75.16
Quat	x	*Squad*	444.57	39.58	506.92
Quat	x	spline on $\mathbb{SE}(3)$	528.46	40.19	591.69

TABLE 6.12: Computation time (in seconds) of our MCBA (rs.c.sfa.int) with 100 iterations for BC2 video using parametrization (Param) for rotation without/with time approximation and camera motion model: linear approximation Eq.(6.14), quadratic approximation Eq.(6.21), *Squad* Eq.(6.23) and spline on $\mathbb{SE}(3)$ Eq.(6.29).

6.6.7 The case close to a singularity

Here we examine the Euler angles involved in our rotation parametrization in Section D.3. Figure 6.10(a) illustrates the Euler angles $(\alpha_i, \beta_i, \gamma_i)$ for keyframe number i for BC1. This function looks continuous (zoom in to see the blue crosses in the electronic version); the largest value of $|\gamma_i - \gamma_{i-1}|$ is equal to 0.536 rad. Such a result is expected since $M(t)$ is assumed to be $\mathcal{C}^3$ continuous and the keyframe sampling t_i is dense enough to obtain a successful SfM result. Furthermore, $|\beta_i|$ is as small as possible to keep away from the singularities $\beta \in \pi/2 + \pi\mathbb{Z}$. According to Table 6.2, all $|\beta_i|$ are less than 0.268 radian, except for FH (all $|\beta_i|$ are less than 0.494). The $|\beta_i|$ RMS is about 0.25-0.47 times the $|\beta_i|$ maximum for every video.

We detail consequences of a naive/unlucky use of Euler angles ignoring singularities (using the notation in Subsection 6.2.3). Assume that the initial multi-camera poses meet $\mathtt{R}_i^0 \approx \mathtt{R}_z(\gamma_i)\mathtt{R}_y(\pi/2)$. This is possible because of a "bad" choice of coordinate system: let rotation $\mathtt{R_A}$ and $\mathtt{R_B}$ as in Appendix D.3 (we have $\forall i$, $\mathtt{R_A}^{-1}\mathtt{R}_i^0\mathtt{R_B}^{-1} \approx \mathtt{R}_z(\gamma_i)$) and rotate the world coordinate system by $\mathtt{R_A}^{-1}$ and the multi-camera coordinate system by $\mathtt{R_B}^{-1}\mathtt{R}_y(\pi/2)$ so that $\mathtt{R}_i^0$ is replaced by $\mathtt{R_A}^{-1}\mathtt{R}_i^0\mathtt{R_B}^{-1}\mathtt{R}_y(\pi/2) \approx \mathtt{R}_z(\gamma_i)\mathtt{R}_y(\pi/2)$. Now we naively set $\mathcal{R} = \mathcal{E}$ and redo the experiments of BC1. Then $\beta_i \approx \pi/2$ (this is very close to a singularity). Angles α_i and γ_i are quite more perturbed (as shown by Figure 6.10(b))



(a) Using Appendix D.3

(b) The case close to singularity

FIGURE 6.10: Rotation parametrization for BC1. x-axis are the keyframe numbers, y-axis are Euler angles.

although they are chosen such that the Euler angle function is as continuous as possible ($\max_i |\gamma_i - \gamma_{i-1}| = 3.099$ and $\max_i |\alpha_i - \alpha_{i-1}| = 3.104$). Using rs.c.sfa.int with linear approximation as Eq.(6.14) and time approximation $t_{\mathbf{p}} = t_{\tilde{\mathbf{p}}}$, the new values of $v_{\max} f\tau$ is 0.7885 and of $f\Delta_j = \{-0.283, -0.149, 0.139\}, j = \{1, 2, 3\}$. Thus the relative error of τ increases (9.7%) in comparison to that in Table 6.4 (0.2%), Table 6.10 and Table 6.11.

6.6.8 Robustness with respect to camera's speed

The previous sections have showed the evaluation of our proposed method. In the whole of our experiments, we use synthetic sequence BC2 (about 18km/h) which mimics bicycle motion in BC1 without motion blur. In this subsection, we reuse the synthetic sequences BC2 in Subsection 5.6.4 such that the mean of the speed is about 2-5 times larger than 18km/h. We also remind that we slow down in sharp turns if the speed is greater than/equal to 20km/h. Table 6.13 shows the results of our method applied to these examples. In comparison with GS and FA approximations in Table 5.7, our MCBA including RS and SFA improves significantly the accuracy in terms of d. We obtain errors $d \leq 1.8$ pixels (expect for the case of 100km/h, $d \approx 4$ pixels) using BA among RS.C.SFA.INT and RS.NC.SFA.INT. Both $e(\tau)$ and $e(\Delta)$ increase when the average speed increases and the speed is greater than 40km/h. We remind that $e(\Delta)$ cumulates absolute errors of SFA synchronization. That means the mean SFA synchronization error of n cameras is about $e(\Delta)/(n-1)$. Since the average speed is known, we convert $e(\Delta)/3$ (for $n = 4$ cameras) in meters: about $[1.6 - 4.0]$ (mm) for speed $v \leq 60$ km/h, 1.3 cm for 80 km/h and 2.7 cm for 100 km/h.

6.7 Conclusion

This chapter introduces the first self-calibration method for a multi-camera moving in a scene, that simultaneously estimates intrinsic and extrinsic parameters, time offsets and line delay for RS in addition to the usual parameters (3D points and multi-camera poses). We experiment in a context that we believe useful for applications (3D modeling and 360 video): several consumer cameras or a spherical camera mounted on a helmet and moving along long trajectories by walking and biking (among others). Long trajectories are useful for calibration accuracy and are allowed since our BA only refines

Speed (v)	kfr	Method	$e(\Delta)$	$\frac{e(\Delta)v}{f(n-1)}$ (m)	$e(\tau)$	d
18km/h (5m/s)	225	gs.c.fa.int				2.018
		gs.nc.fa.int				1.417
		rs.c.sfa.int	0.097	0.0016	2.7%	1.476
		rs.nc.sfa.int	0.110	0.0019	3.7%	0.366
20km/h (5.6m/s)	232	gs.c.fa.int				2.437
		gs.nc.fa.int				1.460
		rs.c.sfa.int	0.131	0.0024	0.6%	1.815
		rs.nc.sfa.int	0.103	0.0019	2.2%	0.902
40km/h (11.1m/s)	211	gs.c.fa.int				3.301
		gs.nc.fa.int				3.708
		rs.c.sfa.int	0.107	0.0040	2.5%	1.395
		rs.nc.sfa.int	0.097	0.0036	1.3%	0.905
60km/h (16.7m/s)	202	gs.c.fa.int				4.789
		gs.nc.fa.int				6.305
		rs.c.sfa.int	0.048	0.0027	6.8%	1.342
		rs.nc.sfa.int	0.043	0.0024	6.2%	1.234
80km/h (22.2m/s)	198	gs.c.fa.int				6.519
		gs.nc.fa.int				7.149
		rs.c.sfa.int	0.181	0.0134	8.3%	1.620
		rs.nc.sfa.int	0.171	0.0126	7.6%	1.595
100km/h (27.8m/s)	182	gs.c.fa.int				9.426
		gs.nc.fa.int				10.189
		rs.c.sfa.int	0.299	0.0277	16.8%	3.726
		rs.nc.sfa.int	0.292	0.0271	16.4%	4.027

TABLE 6.13: SFA time offsets and line delay accuracy for BC2 with respect to multi-camera's speed: $n = 4$ cameras, keyframes kfr and keyframe sampling $N_2 = 900$, $N_3 = 450$ (see Section 2.4.1). Comparison to Table 5.7.

the keyframes provided by SfM. We provide accuracies for calibration/line delay/time offsets with respect to ground truth, examine the influence of the tuning of keyframe selection, show variations of time offsets in a long sequence, experiment and compare different approximations (for time continuous camera trajectory and for reprojection errors), and check the robustness of our method with respect to camera's speed.

Chapter 7

Conclusion

7.1 Summary

In this thesis, we present the first self-calibration method of omnidirectional multi-camera, that simultaneously estimates intrinsic parameters, extrinsic parameters, times offsets between videos and line delay coefficient of rolling shutter in addition to the usual parameters: 3D points and multi-camera poses.

In Chapter 4, we start by a rough calibration thanks to assumptions that are suitable to an omnidirectional multi-camera without a privileged direction: the cameras have the same setting (frequency, image resolution, approximate field-of-view angle) and are roughly equiangular. These initial intrinsic parameters are refined using monocular Structure-from-Motion and bundle adjustment method. Moreover, one of our contributions is the frame-accurate synchronization that deals with cameras without assumption on the field-of-view shared by adjacent cameras. We benefit the important constraint: the cameras are moving jointly. Then, different cameras have the same angular velocity at the same instant. We estimate frame-accurate time offsets based on the instantaneous angular velocity thanks to monocular Structure-from-Motion for all frames. In practice, we do that on few thousands of frames at the beginning of the videos.

In Chapter 5, we initialize the relative transformation between cameras assuming that the multi-camera is roughly central with approximately known inter-camera rotations; and that the cameras are global shutter and frame-accurately synchronized. In our experiments, these assumptions are enough to feed our multi-camera bundle adjustment. In comparison to multi-camera bundle adjustment in [Lébraly+11], we also estimate the intrinsic parameters and minimize the reprojection errors in the original image space (not the rectified one) for the same polynomial distortion model. Our implementation use both the analytical Jacobian matrix and its sparse data structure to achieve an efficient calibration. The extension for the unified camera model is straightforward.

We apply Structure-from-Motion and bundle adjustment in Chapter 5 using simple camera model: global shutter, subframe residual time offsets and central (baseline between cameras are forced to zero). This calibration is used to initialize our bundle adjustment for generalized camera model in Chapter 6 that takes account into subframe-accurate synchronization and line delay coefficient of rolling shutter. This is our main contribution. Some camera motion models are proposed in our work using different

parameterizations for rotations. The simple correction method is used to enforce the continuity of camera motion model. For Euler-based angle parameterization, we develop a technique to keep away from the singularities. We introduce the image projection that takes into account rolling shutter and synchronization. The approximation [Klingner+13] is also experimented in the computation of projection point. Besides, we propose the exact solution for the image projection. Furthermore, our multi-camera bundle adjustment is done over long video datasets (hundreds of meters or kilometers) without additional sensors thanks to keyframe sampling in Structure-from-Motion algorithm and efficient implementation of bundle adjustment step which takes into account sparsity of the Jacobian matrix.

We experiment in cases that we believe useful for applications (3D modeling and 360 videos): several consumer cameras or spherical camera mounted on a helmet and moving along long trajectories by walking and riding. In Chapter 5, we compare to the method in [Lébraly+11] for the polynomial distortion model. We see that minimizing the reprojection errors in the original space is better than that in the rectified space. Moreover, some applications like 360 video needs the central calibration. We use (and build in some cases) central and slightly omnidirectional multi-cameras for these purposes. The central approximation is good if the ratio between camera-scene distance and baseline is large. We also check that the global shutter approximation is less tenable for high speed videos. In Chapter 6 with multi-camera bundle adjustment adding rolling shutter and subframe-accurate synchronization, we provide accuracy for calibration/line delay/time offsets with respect to ground truth. We also examine the influence of the tuning of keyframe selection, check the robustness of SFA refinement with respect to bad FA initialization, show variations of time offsets in a long sequence, experiment and compare different approximations (for time continuous camera trajectory and for reprojection errors) and different parameterizations for rotations. We show the robustness of our proposed method for high speed synthetic videos.

7.2 Future work

Several improvements and future works are possible. In particular, we would like to highlight the following:

- As mentioned in Chapter 4 and 5, we do not intend to compete with the accuracy and generality of previous methods that initialize intrinsic and extrinsic parameters. These initializations can be improved thanks to previous works for both intrinsic parameters and inter-camera poses. And a fully automatic method to estimate these parameters from the environment should be considered, instead of initializing the parameters using our assumption for omnidirectional camera without privileged direction.
- Our Structure-from-Motion steps can fail for several reasons: degenerate camera motion, lack of texture, global shutter approximation and central approximation. Note that Structure-from-Motion is not the contribution of this thesis and previous methods can be applied to avoid the central approximation. Furthermore, a preprocessing should select segment(s) in the video where we safely apply Structure-from-Motion (avoid pure/fast camera rotations, critical cases and blurred/low textured images).

- As we can observe from this thesis, we experiment on long trajectories in urban scene and some of them have many loops. Therefore, our self-calibration method can be improved with a process called loop detection and closure. We detect when the camera passes by an already reconstructed place, perform an image matching between the images of the same point of the scene and perform our multi-camera bundle adjustment with the additional constraints provided by this matching. More generally, we can match points between different cameras and between different keyframes in order to improve the precision.
- Variants of the method can be experimented, e.g. by using alternative keyframe selections and the other camera models.
- We should examine the improvements in applications provided by our non-zero line delay and subframe-accurate time offsets.

Appendix A

Properties of the unified camera model

We remind that this model is described in Chapter 4, Section 4.1.2 using a spherical projection and a perspective projection. Let $\mathbf{k} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^\top$, $\mathbf{c} = \begin{pmatrix} 0 & 0 & -\xi \end{pmatrix}^\top$ and 3D point $\tilde{\mathbf{X}}_\mathrm{C} = (X_\mathrm{C}\ Y_\mathrm{C}\ Z_\mathrm{C})^\top$. Let ν be the angle between $\mathbf{k}$ and a ray of the perspective camera (a half-line started at $\mathbf{c}$ and including $\mathbf{Q}_s = \tilde{\mathbf{X}}_\mathrm{C}/||\tilde{\mathbf{X}}_\mathrm{C}||$). Let μ be the angle between principal direction $\mathbf{k}$ and a ray of the unified camera (half-line started at $\mathbf{0}$ with direction $\mathbf{Q}_s$).

A.1 Theoretical field-of-view

Figure A.1 shows notations ν, μ, μ_0 (a particular value of μ), $\mathbf{Q}_s$, $\mathbf{c}$ and $\mathbf{k}$ in two cases: $\xi > 1$ and $0 < \xi < 1$.

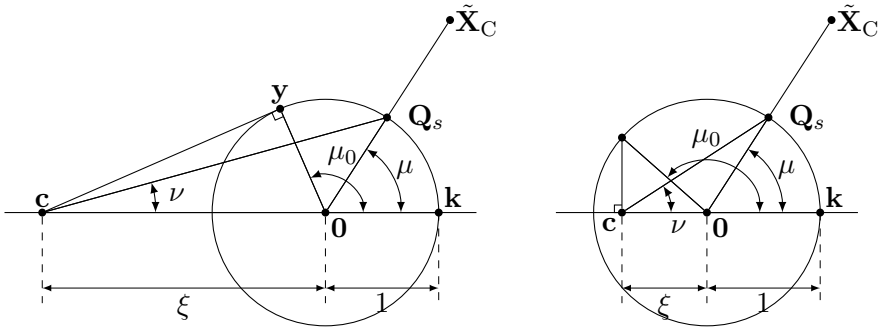


FIGURE A.1: Theoretical FoV in two cases: $\xi > 1$ (left) and $0 < \xi < 1$ (right).

In both cases, μ_0 is the maximum value of μ that ensures that there is only one back-projected ray direction corresponding to the projection $p(\tilde{\mathbf{X}}_\mathrm{C})$ by the unified camera model. The angle μ_0 is the half-angle of the *theoretical* FoV of this model, i.e. by ignoring the bounded size of the image and the projection of the camera itself. The FoV in our work ignores nothing and is included in the theoretical FoV.

If $\xi = 0$, the unified camera model is the standard perspective camera model with center $\mathbf{c} = \mathbf{0}$. Since $\mathbf{Q}_s$ is in front of the perspective camera, the theoretical FoV is the half-space $Z_\mathrm{C} > 0$ and $\mu_0 = \pi/2$.

If $0 < \xi < 1$, $\mathbf{c}$ is inside unit sphere $\mathcal{S}$. Since $\mathbf{Q}_s$ is in front of the perspective camera, the theoretical FoV is the half-space $Z_\mathrm{C} > -\xi$ and $\cos\mu_0 = -\xi$.

If $\xi > 1$, $\mathbf{c}$ is outside $\mathcal{S}$ and $\mathcal{S}$ is entirely in front of the perspective camera. The projection of $\mathcal{S}$ by p is an ellipse and its interior. Let $\mathcal{C}$ be the cone that is tangent to $\mathcal{S}$ with apex $\mathbf{c}$, i.e. the union of every line $(\mathbf{cy})$ that intersects $\mathcal{S}$ at a single point $\mathbf{y}$. We have $\mathbf{y}^\top(\mathbf{c} - \mathbf{y}) = 0$ and $\mathbf{k}^\top\mathbf{y} < 0$ and $\cos\mu_0 = \mathbf{k}^\top\mathbf{y} = 1/(-\xi)$. For example, $\xi = 2$ implies that $\mu \leq \mu_0 = 2\pi/3$.

A.2 Angle of back-projected ray

There is a relation between ν and μ in all cases. Using notation $\begin{pmatrix} X & Y & Z \end{pmatrix}^\top = \mathbf{Q}_s$, we have $\cos\mu = Z$ and $\sin\mu = \sqrt{X^2 + Y^2}$. Since $\mathbf{Q}_s$ is in front of the perspective camera, $\xi + Z > 0$ and $\tan\nu = \dfrac{\sqrt{X^2 + Y^2}}{(\xi + Z)}$. Thus

$$\tan\nu = \frac{\sin\mu}{\xi + \cos\mu}. \tag{A.1}$$

Since $p(\tilde{\mathbf{X}}_\mathrm{C}) = \left(f_x \dfrac{X}{(\xi + Z)} + u_0, f_y \dfrac{Y}{(\xi + Z)} + v_0 \right)$ and $p(\mathbf{k}) = \mathbf{p}_0$,

$$f_x = f_y = f \Rightarrow \frac{||p(\tilde{\mathbf{X}}_\mathrm{C}) - \mathbf{p}_0||}{f} = \tan\nu = \frac{\sin\mu}{\xi + \cos\mu}. \tag{A.2}$$

Appendix B

Analytical Jacobian

In this appendix, we detail the calculation of analytical Jacobian for multi-camera bundle adjustment. First, we remind the similarity transformation from a 3D point in the world coordinate system to a point in the camera coordinate system (using all notations in Section 5.3):

$$\tilde{\mathbf{X}}_{\mathrm{C}} = (\mathtt{R}_{\mathrm{C}}^{j})^{\top} \left[(\mathtt{R}_{\mathrm{M}}^{i})^{\top} \mid -(\mathtt{R}_{\mathrm{M}}^{i})^{\top} \mathbf{t}_{\mathrm{M}}^{i} - \mathbf{t}_{\mathrm{C}}^{j} \right] \mathbf{X}^{l}. \tag{B.1}$$

The forward projection from $\tilde{\mathbf{X}}_{\mathrm{C}}$ to a 2D image point $\mathbf{p}$ (which depends on camera model) is detailed in Section 4.1. So the projected point $\mathbf{p}$ is a function of $\boldsymbol{\theta} = \{\mathbf{X}^{l}, \mathtt{R}_{\mathrm{M}}^{i}, \mathbf{t}_{\mathrm{M}}^{i}, \mathtt{R}_{\mathrm{C}}^{j}, \mathbf{t}_{\mathrm{C}}^{j}, \mathbf{I}^{j}\}$. The observed 2D point is $\tilde{\mathbf{p}}$. We also remind the reprojection error in Section 5.3 as follows

$$\boldsymbol{\epsilon}_{ij}^{l} = \mathbf{p}(\mathbf{X}^{l}, \mathtt{R}_{\mathrm{M}}^{i}, \mathbf{t}_{\mathrm{M}}^{i}, \mathtt{R}_{\mathrm{C}}^{j}, \mathbf{t}_{\mathrm{C}}^{j}, \mathbf{I}^{j}) - \tilde{\mathbf{p}} \tag{B.2}$$

We detail the computation of block matrices in Section 5.3.2 for two cases: the polynomial distortion model and the unified camera model. We also remind that monocular case is a special case of multi-camera case.

B.1 Polynomial distortion model

As discussed in Subsection 4.1.1 and 5.3.1, we do not have closed-form for forward projection. Here we would like to compute $\mathbf{p} = p_j(\tilde{\mathbf{X}}_{\mathrm{C}})$ in Eq.(B.1) and its Jacobian using the camera model in Section 4.1.1 assuming that $\tilde{\mathbf{X}}_{\mathrm{C}}$ is known.

First, we remind the parameter vector and functions g and g_1 such that

$$\mathbf{I}^{j} = (f_x, f_y, \mathbf{p}_0, k_1, k_2, \ldots, k_n),\ \boldsymbol{\theta} = \{\mathbf{X}^{l}, \mathtt{R}_{\mathrm{M}}^{i}, \mathbf{t}_{\mathrm{M}}^{i}, \mathtt{R}_{\mathrm{C}}^{j}, \mathbf{t}_{\mathrm{C}}^{j}, \mathbf{I}^{j}\},\ \mathbf{p}_u = \pi(\mathtt{K}\tilde{\mathbf{X}}_{\mathrm{C}}), \tag{B.3}$$

$$\begin{pmatrix} u \\ v \end{pmatrix} = \mathbf{z},\ \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} = \mathbf{p}_0,\ \bar{r}^2 = \frac{(u-u_0)^2}{f_x^2} + \frac{(v-v_0)^2}{f_y^2},\ \mathtt{K} = \begin{pmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{B.4}$$

$$g(\mathbf{z}, \boldsymbol{\theta}) = \left(1 + \sum_{i=1}^{n} k_i \bar{r}^{2i}\right)(\mathbf{z} - \mathbf{p}_0) - \mathbf{p}_u + \mathbf{p}_0, \tag{B.5a}$$

$$g_1(\mathbf{z}, \boldsymbol{\theta}) = \left(1 + \sum_{i=1}^{n} k_i ||\bar{\mathbf{z}}||^{2i}\right)\bar{\mathbf{z}} - \bar{\mathbf{p}}_u. \tag{B.5b}$$

where $\begin{pmatrix}\bar{\mathbf{z}}\\1\end{pmatrix} = \mathtt{K}^{-1}\begin{pmatrix}\mathbf{z}\\1\end{pmatrix}$ and $\begin{pmatrix}\bar{\mathbf{p}}_u\\1\end{pmatrix} = \mathtt{K}^{-1}\begin{pmatrix}\mathbf{p}_u\\1\end{pmatrix}$. We remind that $\pi(\mathbf{x}) = \pi\left((X\ Y\ Z)^\top\right) = (X/Z\ Y/Z)^\top$.

Second, in Subsection 4.1.1.2, we check that $g(\mathbf{p}, \boldsymbol{\theta}) = \mathbf{0}$. We also note that $\begin{pmatrix}f_x & 0\\0 & f_y\end{pmatrix} g_1 = g$. We obtain $g_1(\mathbf{p}, \boldsymbol{\theta}) = \mathbf{0}$.

Last, using an approximate value $\tilde{\mathbf{p}}$ of $\mathbf{p}$, the projected point $\mathbf{p}$ is estimated by non-linear least-squares minimizing $\mathbf{z} \mapsto ||g(\mathbf{z}, \boldsymbol{\theta}_0)||^2$ where $\boldsymbol{\theta}_0$ is the current value of $\boldsymbol{\theta}$ (provided by initialization or previous iteration of BA). The implicit function Theorem implies that we locally have a $\mathcal{C}^1$ continuous function ψ such that $\mathbf{p} = \psi(\boldsymbol{\theta})$ if $\det \frac{\partial g}{\partial \mathbf{z}} \neq \mathbf{0}$. By differentiating $g(\psi(\boldsymbol{\theta}), \boldsymbol{\theta}) = \mathbf{0}$, we obtain

$$\frac{\partial \mathbf{p}}{\partial \boldsymbol{\theta}} = \frac{\partial \psi}{\partial \boldsymbol{\theta}} = -\left(\frac{\partial g}{\partial \mathbf{z}}\right)^{-1} \frac{\partial g}{\partial \boldsymbol{\theta}}. \tag{B.6}$$

We use Eq.(B.6) for $\mathbf{p}$ derivatives

$$\frac{\partial \mathbf{p}}{\partial k_i} = -\left(\frac{\partial g}{\partial \mathbf{z}}\right)^{-1} \bar{r}^{2i}(\mathbf{p} - \mathbf{p}_0) \tag{B.7}$$

$$\frac{\partial \mathbf{p}}{\partial \tilde{\mathbf{X}}_{\mathrm{C}}} = \left(\frac{\partial g}{\partial \mathbf{z}}\right)^{-1} \frac{\partial \mathbf{p}_u}{\partial \tilde{\mathbf{X}}_{\mathrm{C}}}. \tag{B.8}$$

For derivatives with respect to f_x, f_y, u_0, v_0, we use the function g_1 defined as Eq.(B.5b) and $\mathbf{z} = \mathbf{p}$, $\boldsymbol{\theta} = \boldsymbol{\theta}_0$. We note that

$$\frac{\partial \bar{\mathbf{z}}}{\partial (u_0, v_0)} = -\begin{pmatrix}1/f_x & 0\\0 & 1/f_y\end{pmatrix} = -\frac{\partial \bar{\mathbf{z}}}{\partial \mathbf{z}}, \tag{B.9}$$

$$\frac{\partial \bar{\mathbf{z}}}{\partial (f_x, f_y)} = \frac{\partial \bar{\mathbf{z}}}{\partial \mathbf{z}}\begin{pmatrix}\frac{u_0 - u}{f_x} & 0\\0 & \frac{v_0 - v}{f_y}\end{pmatrix}. \tag{B.10}$$

Thus

$$\frac{\partial g_1}{\partial \mathbf{z}} = \frac{\partial g_1}{\partial \bar{\mathbf{z}}}\frac{\partial \bar{\mathbf{z}}}{\partial \mathbf{z}}, \tag{B.11}$$

$$\frac{\partial g_1}{\partial (u_0, v_0)} = \frac{\partial g_1}{\partial \bar{\mathbf{z}}}\frac{\partial \bar{\mathbf{z}}}{\partial (u_0, v_0)} = -\frac{\partial g_1}{\partial \bar{\mathbf{z}}}\frac{\partial \bar{\mathbf{z}}}{\partial \mathbf{z}} = -\frac{\partial g_1}{\partial \mathbf{z}}, \tag{B.12}$$

$$\frac{\partial g_1}{\partial (f_x, f_y)} = \frac{\partial g_1}{\partial \bar{\mathbf{z}}}\frac{\partial \bar{\mathbf{z}}}{\partial (f_x, f_y)} = \frac{\partial g_1}{\partial \mathbf{z}}\begin{pmatrix}\frac{u_0 - u}{f_x} & 0\\0 & \frac{v_0 - v}{f_y}\end{pmatrix}. \tag{B.13}$$

Thanks to Eq.(B.6), Eq.(B.12) and Eq.(B.13), we obtain

$$\begin{pmatrix} \frac{\partial \mathbf{p}}{\partial f_x} & \frac{\partial \mathbf{p}}{\partial f_y} & \frac{\partial \mathbf{p}}{\partial u_0} & \frac{\partial \mathbf{p}}{\partial v_0} \end{pmatrix} = \begin{pmatrix} \frac{u-u_0}{f_x} & 0 & 1 & 0 \\ 0 & \frac{v-v_0}{f_y} & 0 & 1 \end{pmatrix}. \tag{B.14}$$

We note that the derivative computations in Eq.(B.8) are easy from those of a standard perspective camera (i.e. $\frac{\partial \mathbf{p}_u}{\partial \tilde{\mathbf{X}}_\mathrm{C}}$): multiply on the left side by $\left(\frac{\partial g}{\partial \mathbf{z}}\right)^{-1}$. This also holds for derivatives with respect to all multi-camera parameters thanks to the Chain rule (see Appendix B.3).

B.2 Unified camera model

The forward projection in Subsection 4.1.2 is closed-form and written as

$$p(\tilde{\mathbf{X}}_\mathrm{C}) = \pi\left(\mathtt{K}\left(\frac{\tilde{\mathbf{X}}_\mathrm{C}}{||\tilde{\mathbf{X}}_\mathrm{C}||} + \begin{pmatrix} 0 \\ 0 \\ \xi \end{pmatrix}\right)\right) \tag{B.15a}$$

$$= \pi\left(\mathtt{K}\begin{pmatrix} X_\mathrm{C} \\ Y_\mathrm{C} \\ Z_\mathrm{C} + \rho\xi \end{pmatrix}\right) \tag{B.15b}$$

$$= \begin{pmatrix} f_x \frac{X_\mathrm{C}}{Z_\mathrm{C} + \rho\xi} + u_0 \\ f_y \frac{Y_\mathrm{C}}{Z_\mathrm{C} + \rho\xi} + v_0 \end{pmatrix} \tag{B.15c}$$

where $\rho = ||\tilde{\mathbf{X}}_\mathrm{C}|| = \sqrt{X_\mathrm{C}^2 + Y_\mathrm{C}^2 + Z_\mathrm{C}^2}$. We remind that $\pi(\mathbf{x}) = \pi\left((X\ Y\ Z)^\top\right) = (X/Z\ Y/Z)^\top$.

Thanks to the Chain rule and Eq.(B.15a), we have

$$\frac{\partial \mathbf{p}}{\partial \xi} = \frac{\partial \pi}{\partial \mathbf{x}} \mathtt{K} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{B.16}$$

where

$$\frac{\partial \pi}{\partial \mathbf{x}} = \begin{pmatrix} 1/Z & 0 & -X/Z^2 \\ 0 & 1/Z & -Y/Z^2 \end{pmatrix}. \tag{B.17}$$

Using Eq.(B.15c), the derivative of $\mathbf{p}$ with respect to f_x, f_y, u_0, v_0:

$$\begin{pmatrix} \frac{\partial \mathbf{p}}{\partial f_x} & \frac{\partial \mathbf{p}}{\partial f_y} & \frac{\partial \mathbf{p}}{\partial u_0} & \frac{\partial \mathbf{p}}{\partial v_0} \end{pmatrix} = \begin{pmatrix} \frac{X_\mathrm{C}}{Z_\mathrm{C} + \xi\rho} & 0 & 1 & 0 \\ 0 & \frac{Y_\mathrm{C}}{Z_\mathrm{C} + \xi\rho} & 0 & 1 \end{pmatrix}. \tag{B.18}$$

And using Eq.(B.15b), we obtain

$$\frac{\partial \mathbf{p}}{\partial \tilde{\mathbf{X}}_{\mathrm{C}}} = \frac{\partial \pi}{\partial \mathbf{x}} \mathtt{K} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \xi \dfrac{X_{\mathrm{C}}}{\rho} & \xi \dfrac{Y_{\mathrm{C}}}{\rho} & 1 + \xi \dfrac{Z_{\mathrm{C}}}{\rho} \end{pmatrix}. \tag{B.19}$$

Finally, from Eq.(B.1) and Eq.(B.19), using the chain rule, we obtain the derivative of $\mathbf{p}$ with respect to $\{\mathbf{X}^l, \mathtt{R}_{\mathrm{M}}^i(\mathbf{r}_{\mathrm{M}}^i), \mathbf{t}_{\mathrm{M}}^i, \mathtt{R}_{\mathrm{C}}^j(\mathbf{r}_{\mathrm{C}}^j), \mathbf{t}_{\mathrm{C}}^j\}$ (see Appendix B.3).

B.3 Derivatives with respect to all multi-camera parameters

The Chain rule leads to

$$\frac{\partial \mathbf{p}}{\partial \boldsymbol{\theta}} = \frac{\partial \mathbf{p}}{\partial \tilde{\mathbf{X}}_{\mathrm{C}}} \frac{\partial \tilde{\mathbf{X}}_{\mathrm{C}}}{\partial \boldsymbol{\theta}} \tag{B.20}$$

where $\tilde{\mathbf{X}}_{\mathrm{C}}$ is computed as in Eq.(B.1) and

(a) Partial derivative with respect to 3D points $\mathcal{X} = \{\mathbf{X}^l\}$

$$\frac{\partial \tilde{\mathbf{X}}_{\mathrm{C}}}{\partial \mathbf{X}^l} = (\mathtt{R}_{\mathrm{C}}^j)^\top \left[(\mathtt{R}_{\mathrm{M}}^i)^\top \mid -(\mathtt{R}_{\mathrm{M}}^i)^\top \mathbf{t}_{\mathrm{M}}^i - \mathbf{t}_{\mathrm{C}}^j \right] \tag{B.21}$$

(b) Partial derivative with respect to multi-camera poses

$$\frac{\partial \tilde{\mathbf{X}}_{\mathrm{C}}}{\partial \mathbf{r}_{\mathrm{M}}^i} = (\mathtt{R}_{\mathrm{C}}^j)^\top \frac{\partial (\mathtt{R}_{\mathrm{M}}^i)^\top}{\partial \mathbf{r}_{\mathrm{M}}^i} \left[\mathtt{I} \mid -\mathbf{t}_{\mathrm{M}}^i \right] \mathbf{X}^l \tag{B.22}$$

$$\frac{\partial \tilde{\mathbf{X}}_{\mathrm{C}}}{\partial \mathbf{t}_{\mathrm{M}}^i} = -w(\mathtt{R}_{\mathrm{C}}^j)^\top (\mathtt{R}_{\mathrm{M}}^i)^\top \tag{B.23}$$

(c) Partial derivative with respect to extrinsic parameters

$$\frac{\partial \tilde{\mathbf{X}}_{\mathrm{C}}}{\partial \mathbf{r}_{\mathrm{C}}^j} = \frac{\partial (\mathtt{R}_{\mathrm{C}}^j)^\top}{\partial \mathbf{r}_{\mathrm{C}}^j} \left[(\mathtt{R}_{\mathrm{M}}^i)^\top \mid -(\mathtt{R}_{\mathrm{M}}^i)^\top \mathbf{t}_{\mathrm{M}}^i - \mathbf{t}_{\mathrm{C}}^j \right] \mathbf{X}^l \tag{B.24}$$

$$\frac{\partial \tilde{\mathbf{X}}_{\mathrm{C}}}{\partial \mathbf{t}_{\mathrm{C}}^j} = -w(\mathtt{R}_{\mathrm{C}}^j)^\top. \tag{B.25}$$

Appendix C

Minimal parametrization

We remind the term *gauge freedom* in [Triggs+00], [Hartley+04]. The work gauge just means reference coordinate for a parameter set and gauge freedom refers to the freedom in the choice of reference coordinate of the parameter set, without affecting the underlying geometry. Hence, gauge freedom does not essentially affect the cost function. Consider the choice of 3D coordinate system in reconstruction problems, BA updates can perturb reconstructed scene almost arbitrarily. It causes the normal equation Eqs.(2.27) and (2.28) to be singular (the matrix is not invertible) and hence to have multiple solutions. LM algorithm leads to slower convergence in evidence if there is excessive gauge freedom.

Besides 3D coordinate system, many other types of geometric parametrization involve arbitrary choices. These include homogeneous vectors for 3D points, quaternions, etc. In this appendix, we detail some methods for obtaining minimal parameterizations for some parameter sets: 3D coordinate system (Appendix C.1) and homogeneous vectors (Appendix C.2).

C.1 Parametrization of 3D coordinate system

C.1.1 General case

Let $\Theta \in \mathbb{R}^n$, $\Psi \in \mathbb{R}^m$ and $E : \mathbb{R}^n \to \mathbb{R}^m$ a $\mathcal{C}^2$ continuous function such that $m \geq n$. We would like to find the minimizer Θ of function $F(\Theta) = \frac{1}{2}||E(\Theta) - \Psi||^2$ for a given Ψ. Let $\mathtt{J} = \frac{\partial E}{\partial \Theta} \in \mathbb{R}^{m \times n}$ and $k = \dim \ker \mathtt{J} < n$. Note that $\ker \mathtt{J}$ and $\operatorname{im} \mathtt{J}$ are the kernel and image of matrix $\mathtt{J}$.

We assume that k is constant for all Θ ($\mathtt{J}$ is a function of Θ). If $k = 0$, Θ is a minimal parametrization for E. If $k \geq 1$ and $\Theta_0 \in \mathbb{R}^n$, we assume that there is a $\mathcal{C}^1$ continuous function $S_{\Theta_0} : \mathbb{R}^k \to \mathbb{R}^n$ such that $S_{\Theta_0}(\mathbb{R}^k) = \{\Theta \in \mathbb{R}^n, E(\Theta) = E(\Theta_0)\}$ and $\ker \frac{\partial S_{\Theta_0}}{\partial \vartheta} = 0$, $\vartheta \in \mathbb{R}^k$.

Lemma C.1. *If $k \geq 1$, $\operatorname{im} \frac{\partial S_{\Theta_0}}{\partial \vartheta} = \ker \mathtt{J}$. (Here, we have $\mathtt{J} = \mathtt{J}(S_{\Theta_0}(\vartheta))$.)*

Proof. The derivative of $E(S_{\Theta_0}(\vartheta)) = E(\Theta_0)$ is computed by the chain rule: $\mathtt{J}\frac{\partial S_{\Theta_0}}{\partial \vartheta} = \mathbf{0}$. Thus $\operatorname{im} \frac{\partial S_{\Theta_0}}{\partial \vartheta} \subseteq \ker \mathtt{J}$. Since

$$\dim \operatorname{im} \frac{\partial S_{\Theta_0}}{\partial \vartheta} = k - \dim \ker \frac{\partial S_{\Theta_0}}{\partial \vartheta} = k = \dim \ker \mathtt{J}, \tag{C.1}$$

we obtain the result. □

In the case of $k \geq 1$,

- Θ is an over-parametrization for E
- S_{Θ_0} has k DoF; E has $n - k$ DoF
- intuitively, there are k coefficient(s) in Θ that are useless for the minimization (Θ is not a minimal parametrization for E)
- LM algorithm deals with this over-parametrization if k is "small".

C.1.2 Monocular case

Here Ψ is the vector concatenation of all 2D (inlier) detected points, Θ is the vector of concatenation of

- poses $(\mathtt{R}_i, \mathbf{t}_i)$ at the i-th key frame,
- 3D points $\tilde{\mathbf{X}}_l$ (it is more convenient to use inhomogeneous vector),
- optional intrinsic parameters,
- optional line delay for RS camera.

Let $s > 0$, $\mathtt{R}$ be a rotation, $\mathbf{t} \in \mathbb{R}^3$ be a translation. We define similarity transformation in $\mathbb{R}^3$ by

$$T_{(s,\mathtt{R},\mathbf{t})}(\mathbf{z}) = s\mathtt{R}^\top(\mathbf{z} - \mathbf{t}) \tag{C.2}$$

where $\mathbf{z} \in \mathbb{R}^3$. Note that $T^{-1}_{(s,\mathtt{R},\mathbf{t})} = T_{(1/s,\mathtt{R}^\top,-s\mathtt{R}^\top\mathbf{t})}$. Our $E(\Theta)$ is the concatenation of all projection function $\pi\left(T_{(1,\mathtt{R}_i,\mathbf{t}_i)}(\tilde{\mathbf{X}}_l)\right)$. The function π meets $\pi(\lambda\mathbf{x}) = \pi(\mathbf{x})$ where $\mathbf{x} \in \mathbb{R}^3 \setminus \{\mathbf{0}\}$ and $\lambda > 0$. The function E can also have other parameters: optional intrinsic and rolling shutter parameters.

C.1.2.1 Over-parametrization

Now we define $S_\Theta(s, \mathtt{R}, \mathbf{t})$ by a modified version of Θ as follows:

- all $\mathtt{R}_i$ of Θ are replaced by $\mathtt{R}^\top\mathtt{R}_i$,
- all $\mathbf{t}_i$ of Θ are replaced by $T_{(s,\mathtt{R},\mathbf{t})}(\mathbf{t}_i)$,
- all $\tilde{\mathbf{X}}_l$ of Θ are replaced by by $T_{(s,\mathtt{R},\mathbf{t})}(\tilde{\mathbf{X}}_l)$,

- the other (optional intrinsic or rolling shutter) parameters of Θ are unchanged.

Since

$$\begin{aligned} T_{\left(1,\mathtt{R}^\top \mathtt{R}_i, T_{(s,\mathtt{R},\mathbf{t})}(\mathbf{t}_i)\right)}\left(T_{(s,\mathtt{R},\mathbf{t})}(\tilde{\mathbf{X}}_l)\right) &= (\mathtt{R}^\top \mathtt{R}_i)^\top \left(s\mathtt{R}^\top(\tilde{\mathbf{X}}_l - \mathbf{t}) - s\mathtt{R}^\top(\mathbf{t}_i - \mathbf{t})\right) \\ &= s\mathtt{R}_i^\top(\tilde{\mathbf{X}}_l - \mathbf{t}_i) = sT_{(1,\mathtt{R}_i,\mathbf{t}_i)}(\tilde{\mathbf{X}}_l) \end{aligned} \tag{C.3}$$

and $\pi(\lambda\mathbf{x}) = \pi(\mathbf{x})$, we have

$$\pi\left(T_{\left(1,\mathtt{R}^\top \mathtt{R}_i, T_{(s,\mathtt{R},\mathbf{t})}(\mathbf{t}_i)\right)}\left(T_{(s,\mathtt{R},\mathbf{t})}(\tilde{\mathbf{X}}_l)\right)\right) = \pi\left(T_{(1,\mathtt{R}_i,\mathbf{t}_i)}(\tilde{\mathbf{X}}_l)\right). \tag{C.4}$$

By concatenating, we obtain

$$E\left(S_\Theta(s,\mathtt{R},\mathbf{t})\right) = E(\Theta). \tag{C.5}$$

Note that S_Θ is a world coordinate change by similarity transformation $T_{(s,\mathtt{R},\mathbf{t})}$ of reconstruction Θ. Now, we assume that all assertions in Appendix C.1.1 are true for S_Θ and $k = 1+3+3 = 7$ (a rotation is counted as 3 parameters); Θ is an over-parametrization for E.

C.1.2.2 Minimal parametrization

We need to fix $k = 7$ independent parameters to obtain a minimal parametrization, e.g. we remove from Θ the pose $(\mathtt{R}_0, \mathbf{t}_0)$ and one of the 3 coordinates of one $\mathbf{t}_i$ where $i \neq 0$ and which is not equal to that of $\mathbf{t}_0$. Removing $\mathtt{R}_0$ from Θ locks $\mathtt{R}$ in $S_\Theta(s,\mathtt{R},\mathbf{t})$ since the projections change in the 0-th image if $\mathtt{R} \neq \mathtt{I}_3$ (identity matrix). Similarly, removing $\mathbf{t}_0$ from Θ locks $\mathbf{t}$ in $S_\Theta(s,\mathtt{R},\mathbf{t})$ since the projections change in the 0-th image if $\mathbf{t} \neq \mathbf{0}$. Removing two same coordinates of $\mathbf{t}_0$ and $\mathbf{t}_1$ from Θ locks s in $S_\Theta(s,\mathtt{R},\mathbf{t})$.

C.1.3 Multi-camera case

Here Ψ is the vector concatenation of all 2D (inlier) detected points, Θ is the vector of concatenation of

- multi-camera poses $(\mathtt{R}_\mathrm{M}^i, \mathbf{t}_\mathrm{M}^i)$ at the i-th key frame,
- 3D points $\tilde{\mathbf{X}}^l$,
- poses $(\mathtt{R}_\mathrm{C}^j, \mathbf{t}_\mathrm{C}^j)$ of the j-th camera in the multi-camera coordinate system; $\mathbf{t}_\mathrm{C}^j$ is optional if the multi-camera is central ($\mathbf{t}_\mathrm{C}^j = \mathbf{0}$ in this case),
- optional intrinsic parameters for each camera,
- optional time offset Δ_j of the j-th camera,
- optional line delay for RS camera.

Furthermore, E is the concatenation of all projection functions

$$\pi\left(T_{(1,\mathtt{R}_\mathrm{C}^j,\mathbf{t}_\mathrm{C}^j)}\left(T_{(1,\mathtt{R}_\mathrm{M}^i,\mathbf{t}_\mathrm{M}^i)}(\tilde{\mathbf{X}}^l)\right)\right). \tag{C.6}$$

Function π has the same properties as in Appendix C.1.2.

C.1.3.1 Over-parametrization

For several reasons, Θ is an over-parametrization for E. According to Eq.(C.3), we have

$$T_{\left(1,\mathtt{R}^\top \mathtt{R}^i_{\mathrm{M}},T_{(s,\mathtt{R},\mathbf{t})}(\mathbf{t}^i_{\mathrm{M}})\right)}\left(T_{(s,\mathtt{R},\mathbf{t})}(\tilde{\mathbf{X}}^l)\right) = sT_{(1,\mathtt{R}^i_{\mathrm{M}},\mathbf{t}^i_{\mathrm{M}})}(\tilde{\mathbf{X}}^l). \tag{C.7}$$

We also have

$$T_{(1,\mathtt{R}^j_{\mathrm{C}},s\mathbf{t}^j_{\mathrm{C}})}\left(sT_{(1,\mathtt{R}^i_{\mathrm{M}},\mathbf{t}^i_{\mathrm{M}})}(\tilde{\mathbf{X}}^l)\right) = sT_{(1,\mathtt{R}^j_{\mathrm{C}},\mathbf{t}^j_{\mathrm{C}})}\left(T_{(1,\mathtt{R}^i_{\mathrm{M}},\mathbf{t}^i_{\mathrm{M}})}(\tilde{\mathbf{X}}^l)\right). \tag{C.8}$$

Thus we have $E\left(S^a_\Theta(s,\mathtt{R},\mathbf{t})\right) = E(\Theta)$ if S^a_Θ is a modified version of Θ as follows:

- all $\mathtt{R}^i_{\mathrm{M}}$ of Θ are replaced by $\mathtt{R}^\top \mathtt{R}^i_{\mathrm{M}}$,
- all $\mathbf{t}^i_{\mathrm{M}}$ of Θ are replaced by $T_{(s,\mathtt{R},\mathbf{t})}(\mathbf{t}^i_{\mathrm{M}})$,
- all $\tilde{\mathbf{X}}^l$ of Θ are replaced by $T_{(s,\mathtt{R},\mathbf{t})}(\tilde{\mathbf{X}}^l)$,
- all $\mathbf{t}^j_{\mathrm{C}}$ are replaced by $s\mathbf{t}^j_{\mathrm{C}}$,
- the other ($\mathtt{R}^j_{\mathrm{C}}$, optional intrinsic, time offset and rolling shutter) parameters of Θ are unchanged.

Note that S^a_Θ is similar to S_Θ of the monocular case; the differences are the update of $\mathbf{t}^j_{\mathrm{C}}$.

First, we have

$$\begin{aligned} T_{(1,\mathtt{R}^j_{\mathrm{C}},\mathbf{t}^j_{\mathrm{C}})}\left(T_{(1,\mathtt{R}^i_{\mathrm{M}},\mathbf{t}^i_{\mathrm{M}})}(\tilde{\mathbf{X}}^l)\right) &= (\mathtt{R}^j_{\mathrm{C}})^\top\left((\mathtt{R}^i_{\mathrm{M}})^\top(\tilde{\mathbf{X}}^l - \mathbf{t}^i_{\mathrm{M}}) - \mathbf{t}^j_{\mathrm{C}}\right) \\ &= (\mathtt{R}^i_{\mathrm{M}}\mathtt{R}^j_{\mathrm{C}})^\top(\tilde{\mathbf{X}}^l - \mathbf{t}^i_{\mathrm{M}}) - (\mathtt{R}^j_{\mathrm{C}})^\top\mathbf{t}^j_{\mathrm{C}}. \end{aligned} \tag{C.9}$$

Thus

$$T_{(1,\mathtt{R}^j_{\mathrm{C}},\mathbf{t}^j_{\mathrm{C}})}\left(T_{(1,\mathtt{R}^i_{\mathrm{M}},\mathbf{t}^i_{\mathrm{M}})}(\tilde{\mathbf{X}}^l)\right) = T_{(1,\mathtt{R}^\top\mathtt{R}^j_{\mathrm{C}},\mathtt{R}^\top\mathbf{t}^j_{\mathrm{C}})}\left(T_{(1,\mathtt{R}^i_{\mathrm{M}}\mathtt{R},\mathbf{t}^i_{\mathrm{M}})}(\tilde{\mathbf{X}}^l)\right). \tag{C.10}$$

Second, we define $S^b_\Theta(\mathtt{R})$ as a modified version of Θ as follows. It has the same components with the following exception:

- $\mathtt{R}^j_{\mathrm{C}}$ are replaced by $\mathtt{R}^\top\mathtt{R}^j_{\mathrm{C}}$,
- $\mathbf{t}^j_{\mathrm{C}}$ are replaced by $\mathtt{R}^\top\mathbf{t}^j_{\mathrm{C}}$ and
- $\mathtt{R}^i_{\mathrm{M}}$ are replaced by $\mathtt{R}^i_{\mathrm{M}}\mathtt{R}$.

Thanks to Eqs.(C.10) and (C.6), $E(S^b_\Theta(\mathtt{R})) = E(\Theta)$.

Let $\tilde{\mathbf{t}}^i_{\mathrm{M}}$ and $\tilde{\mathbf{t}}^j_{\mathrm{C}}$ be in $\mathbb{R}^3$ such that $\mathbf{0} = (\mathtt{R}^i_{\mathrm{M}}\mathtt{R}^j_{\mathrm{C}})^\top\tilde{\mathbf{t}}^i_{\mathrm{M}} + (\mathtt{R}^j_{\mathrm{C}})^\top\tilde{\mathbf{t}}^j_{\mathrm{C}}$ for all i,j. Thus Eq.(C.9) implies

$$T_{(1,\mathtt{R}^j_{\mathrm{C}},\mathbf{t}^j_{\mathrm{C}})}\left(T_{(1,\mathtt{R}^i_{\mathrm{M}},\mathbf{t}^i_{\mathrm{M}})}(\tilde{\mathbf{X}}^l)\right) = T_{(1,\mathtt{R}^j_{\mathrm{C}},\mathbf{t}^j_{\mathrm{C}}+\tilde{\mathbf{t}}^j_{\mathrm{C}})}\left(T_{(1,\mathtt{R}^i_{\mathrm{M}},\mathbf{t}^i_{\mathrm{M}}+\tilde{\mathbf{t}}^i_{\mathrm{M}})}(\tilde{\mathbf{X}}^l)\right). \tag{C.11}$$

Since $\mathbf{0} = (\mathtt{R}^i_{\mathrm{M}})^\top \tilde{\mathbf{t}}^i_{\mathrm{M}} + \tilde{\mathbf{t}}^j_{\mathrm{C}}$, all $\tilde{\mathbf{t}}^j_{\mathrm{C}}$ are equal and $\tilde{\mathbf{t}}^i_{\mathrm{M}} = -\mathtt{R}^i_{\mathrm{M}}\tilde{\mathbf{t}}^0_{\mathrm{C}}$. Thus

$$T_{(1,\mathtt{R}^j_{\mathrm{C}},\mathbf{t}^i_{\mathrm{C}})}\left(T_{(1,\mathtt{R}^i_{\mathrm{M}},\mathbf{t}^i_{\mathrm{M}})}(\tilde{\mathbf{X}}^l)\right) = T_{(1,\mathtt{R}^j_{\mathrm{C}},\mathbf{t}^j_{\mathrm{C}}+\mathbf{t})}\left(T_{(1,\mathtt{R}^i_{\mathrm{M}},\mathbf{t}^i_{\mathrm{M}}-\mathtt{R}^i_{\mathrm{M}}\mathbf{t})}(\tilde{\mathbf{X}}^l)\right). \tag{C.12}$$

We define $S^c_\Theta(\mathbf{t})$ as a modified version of Θ as follows. It has the same components with the following exceptions:

- $\mathbf{t}^j_{\mathrm{C}}$ is replaced by $\mathbf{t}^j_{\mathrm{C}} + \mathbf{t}$ and
- $\mathbf{t}^i_{\mathrm{M}}$ is replaced by $\mathbf{t}^i_{\mathrm{M}} - \mathtt{R}^i_{\mathrm{M}}\mathbf{t}$.

Thanks to Eqs.(C.12) and (C.6), $E(S^c_\Theta) = E(\Theta)$.

Last, we combine S^a_Θ, S^b_Θ and S^c_Θ above into a single S_Θ for two cases: the multi-camera is central (i.e. $\mathbf{t}^j_{\mathrm{C}}$ is not a parameters vector of Θ) or non-central ($\mathbf{t}^j_{\mathrm{C}}$ is a parameters vector of Θ). In the central case,

$$S_\Theta(s_a, \mathtt{R}_a, \mathbf{t}_a, \mathtt{R}_b) = S^a_{S^b_\Theta(\mathtt{R}_b)}(s_a, \mathtt{R}_a, \mathbf{t}_a) \tag{C.13}$$

and we assume that all assertions in Appendix C.1.1 are true for S_Θ and $k = 1{+}3{+}3{+}3 = 10$. In the non-central case,

$$S_\Theta(s_a, \mathtt{R}_a, \mathbf{t}_a, \mathtt{R}_b, \mathbf{t}_c) = S^a_{S^b_{S^c_\Theta(\mathbf{t}_c)}(\mathtt{R}_b)}(s_a, \mathtt{R}_a, \mathbf{t}_a) \tag{C.14}$$

and we assume that all assertions in Appendix C.1.1 are true for S_Θ and $k = 1 + 3 + 3 + 3 + 3 = 13$.

C.1.3.2 Minimal parametrization

We need to fix $k = 10$ (central case) or $k = 13$ (non-central case) independent parameters to obtain a minimal parametrization. Here is one example. As in the monocular case, we remove from Θ the pose $(\mathtt{R}^0_{\mathrm{M}}, \mathbf{t}^0_{\mathrm{M}})$ and one of the 3 coordinates of one $\mathbf{t}^i_{\mathrm{M}}$ where $i \neq 0$ and which is not equal to that of $\mathbf{t}^0_{\mathrm{M}}$. Then we remove $\mathtt{R}^0_{\mathrm{C}}$. In the non-central case, we also remove $\mathbf{t}^0_{\mathrm{C}}$.

C.2 Parametrizations of homogeneous vectors

Let $\mathbf{x} \in \mathbb{R}^n$ be an over-parametrization such as homogeneous 4-vector for 3D points and rotations (unit quaternion). It is numerically and computationally more effective to optimize $\mathbf{x}$ over the tangential hyperplane (of the unit sphere $\mathbb{S}^{n-1}$) using $\Delta\mathbf{x} \in \mathbb{R}^{n-1}$ at point $\mathbf{x} \in \mathbb{S}^{n-1}$, then "move" to the point $\mathbf{x}'$. We define a function

$$\mathbf{x}' = \boxplus(\mathbf{x}, \Delta\mathbf{x}) \tag{C.15}$$

where $\mathbf{x}'$ has the same size of $\mathbf{x}$. The $\boxplus$ operator is a generalization of the vector addition.

C.2.1 Parametrization of 3D points

Homogeneous n-vector is used for 3D points ($n = 4$) instead of an Euclidean vector, because it can represent points at infinity. In BA context, it is useful to update orthogonally to that n-vector [Hartley+04][Appendix 6.9.3 and check Corrections and Errata]. Assume that last element of $\mathbf{x}$ is the scalar component of the homogeneous vector. Let $\Delta\mathbf{x}$ be a $n-1$-vector and define $\boxplus$ to be

$$\boxplus(\mathbf{x}, \Delta\mathbf{x}) = ||\mathbf{x}||\mathtt{H}_{\mathbf{v}(\mathbf{x})} f(\Delta\mathbf{x}) \tag{C.16}$$

where $f(\Delta\mathbf{x}) = \left(\text{sinc}(0.5||\Delta\mathbf{x}||)0.5\Delta\mathbf{x}^\top, \cos(0.5||\Delta\mathbf{x}||)\right)^\top$ and $\mathtt{H}_{\mathbf{v}(\mathbf{x})}$ is Householder matrix such that $\mathtt{H}_{\mathbf{v}(\mathbf{x})}\mathbf{x} = ||\mathbf{x}||(0, \ldots, 0, 1)^\top$.

C.2.2 Parametrization of unit quaternions

The quaternion representation of a rotation (see Section 2.2.4) is a redundant representation in that it has one extra DoF. [Agarwal+] proposes an algebraic approach for unit quaternions. Nonlinear optimization moves $\mathbf{x}$ to a new point $\mathbf{x}'$ on the unit sphere. The quaternion difference between these two quaternions is

$$\Delta\mathbf{q} = \mathbf{x}'\mathbf{x}^{-1} \tag{C.17}$$

where the multiplication between two quaternions is defined in Appendix E. $\Delta\mathbf{q}$ is a unit quaternion since $\mathbf{x}$ and $\mathbf{x}'$ are unit quaternions (thanks to Prop.E.2). Thanks to Prop.E.4, there exists a vector $\mathbf{v} \in \mathbb{R}^3, ||\mathbf{v}|| = 1$ and $\theta \in \mathbb{R}$ such that $\Delta\mathbf{q} = [\cos(\theta), \sin(\theta)\mathbf{v}]$. Note that θ is not necessarily limited to interval $]-\pi, \pi]$ since θ and $\theta + 2\pi\mathbb{Z}$ represent a same quaternion. The vector $\Delta\mathbf{x} \in \mathbb{R}^3$ is defined by $\Delta\mathbf{x} = \theta\mathbf{v}$. The $\boxplus$ operator is defined by the multiplication between two quaternions

$$\boxplus(\mathbf{x}, \Delta\mathbf{x}) = [\cos(||\Delta\mathbf{x}||), \text{sinc}(||\Delta\mathbf{x}||)\Delta\mathbf{x}]\,\mathbf{x}. \tag{C.18}$$

It is preferred to use sinc function since sinc is well-defined at zero.

Appendix D

Euler parametrization

In this appendix, we detail the singularities of Euler angles representation and explain how to avoid these singularities in our context. Let $\mathcal{E}(\alpha, \beta, \gamma) = \mathtt{R}_z(\gamma)\mathtt{R}_y(\beta)\mathtt{R}_x(\alpha)$ where $\mathtt{R}_x(\alpha), \mathtt{R}_y(\beta)$ and $\mathtt{R}_z(\gamma)$ are the rotations around the vectors of the canonical basis of $\mathbb{R}^3$ with respective to angles α, β and γ. We use shortened notations $c_\alpha = \cos\alpha, s_\alpha = \sin\alpha$, $c_\beta = \cos\beta, s_\beta = \sin\beta$, $c_\gamma = \cos\gamma, s_\gamma = \sin\gamma$. Thus

$$\mathcal{E}(\alpha, \beta, \gamma) = \begin{bmatrix} c_\gamma c_\beta & c_\gamma s_\beta s_\alpha - s_\gamma c_\alpha & c_\gamma s_\beta c_\alpha + s_\gamma s_\alpha \\ s_\gamma c_\beta & s_\gamma s_\beta s_\alpha + c_\gamma c_\alpha & s_\gamma s_\beta c_\alpha - c_\gamma s_\alpha \\ -s_\beta & c_\beta s_\alpha & c_\beta c_\alpha \end{bmatrix}. \tag{D.1}$$

D.1 Rotation matrix to Euler angle conversion

A general rotation matrix has the form

$$\mathtt{R} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}. \tag{D.2}$$

Given a rotation matrix $\mathtt{R}$, we can compute the Euler angle (α, β, γ) such that $\mathcal{E}(\alpha, \beta, \gamma) = \mathtt{R}$ using a pseudo-code implementation as following:

if $R_{31} \neq \pm 1$ **then**
  $\beta = -\arcsin(R_{31})$; or $\beta = \pi + \arcsin(R_{31})$;
  $\alpha = \arctan2\left(\frac{R_{32}}{\cos\beta}, \frac{R_{33}}{\cos\beta}\right)$;
  $\gamma = \arctan2\left(\frac{R_{21}}{\cos\beta}, \frac{R_{11}}{\cos\beta}\right)$;
else
  $\gamma =$ anything; {can be set to 0}
  if $R_{31} = -1$ **then**
    $\beta = \pi/2$; $\quad \alpha = \gamma + \arctan2(R_{12}, R_{13})$
  else
    $\beta = -\pi/2$; $\quad \alpha = -\gamma + \arctan2(-R_{12}, -R_{13})$
  end if
end if

D.2 The singularities of Euler angles

Let $\partial\mathcal{E}$ be the Jacobian of $\mathcal{E}$ at (α, β, γ) with respect to (α, β, γ). We have

$$\partial\mathcal{E} = \begin{pmatrix} \frac{\partial\mathcal{E}}{\partial\alpha} & \frac{\partial\mathcal{E}}{\partial\beta} & \frac{\partial\mathcal{E}}{\partial\gamma} \end{pmatrix} \in \mathbb{R}^{9\times 3}. \tag{D.3}$$

Lemma D.1. $\ker \partial\mathcal{E} \neq 0$ *if and only if there is* $k \in \mathbb{Z}$ *such that* $\beta = \pi/2 + k\pi$*, i.e. if and only if the coefficient on the bottom-left corner of* $\mathcal{E}(\alpha, \beta, \gamma)$ *is* 1 *or* -1*.*

Proof. First we show that $\beta = \pi/2 + k\pi$ implies $\ker\partial\mathcal{E} \neq 0$. Let $\varepsilon = 1$ if k is even, otherwise $\varepsilon = -1$. We have $s_\beta = \sin(\varepsilon\pi/2) = \varepsilon, c_\beta = \cos(\varepsilon\pi/2) = 0, \sin(\varepsilon\gamma) = \varepsilon s_\gamma$ and $\cos(\varepsilon\gamma) = c_\gamma$. Thus

$$\begin{aligned} \mathcal{E}(\alpha, \varepsilon\pi/2, \gamma) &= \begin{bmatrix} 0 & \varepsilon c_\gamma s_\alpha - s_\gamma c_\alpha & \varepsilon c_\gamma c_\alpha + s_\gamma s_\alpha \\ 0 & \varepsilon s_\gamma s_\alpha + c_\gamma c_\alpha & \varepsilon s_\gamma c_\alpha - c_\gamma s_\alpha \\ -\varepsilon & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 & \varepsilon\sin(\alpha - \varepsilon\gamma) & \varepsilon\cos(\alpha - \varepsilon\gamma) \\ 0 & \cos(\alpha - \varepsilon\gamma) & -\sin(\alpha - \varepsilon\gamma) \\ -\varepsilon & 0 & 0 \end{bmatrix}. \end{aligned} \tag{D.4}$$

Thus $\delta \mapsto \mathcal{E}(\alpha + \varepsilon\delta, \varepsilon\pi/2, \gamma + \delta)$ is a constant function. We derivate it at $\delta = 0$ thanks to the chain rule and obtain $\begin{pmatrix} \frac{\partial\mathcal{E}}{\partial\alpha} & \frac{\partial\mathcal{E}}{\partial\beta} & \frac{\partial\mathcal{E}}{\partial\gamma} \end{pmatrix} \begin{pmatrix} \varepsilon & 0 & 1 \end{pmatrix}^\top = \mathbf{0}$ at point (α, β, γ).

Second we show that $\beta \neq \pi/2 + k\pi$ and $\begin{pmatrix} \frac{\partial\mathcal{E}}{\partial\alpha} & \frac{\partial\mathcal{E}}{\partial\beta} & \frac{\partial\mathcal{E}}{\partial\gamma} \end{pmatrix} \begin{pmatrix} a & b & c \end{pmatrix}^\top = \mathbf{0}$ imply $a = b = c = 0$. Using derivate of the first column of Eq.(D.1), we obtain

$$b \begin{pmatrix} -c_\gamma s_\beta \\ -s_\gamma s_\beta \\ -c_\beta \end{pmatrix} + c \begin{pmatrix} -s_\gamma c_\beta \\ c_\gamma c_\beta \\ 0 \end{pmatrix} = \mathbf{0} \text{ and } c_\beta \neq 0. \tag{D.5}$$

Thus $b = 0$ and $c = 0$. Now we have $a\frac{\partial\mathcal{E}}{\partial\alpha} = \mathbf{0}$. Using derivative of the last row of Eq.(D.1) and $c_\beta \neq 0$, we obtain $a = 0$. □

D.3 Keep away from the singularities

Here we not only compute $\mathtt{R_A}$ and $\mathtt{R_B}$ in the definition of our rotation parametrization $\mathcal{R}$ (Eq.(6.4)), but initialize $\mathbf{r}_\mathrm{M}(t_i)$. Let $\mathtt{R}_i^0$ be the rotation of the i-th keyframe estimated by GS BA (before the final BA in Chapter 6). Using the definitions of $\mathbf{r}_\mathrm{M}$ and t_i in Section 6.2, $\mathbf{r}_\mathrm{M}(t_i)$ is initialized such that $\mathcal{R}(\mathbf{r}_\mathrm{M}(t_i)) = \mathtt{R}_i^0$. We also check that $\mathbf{r}_\mathrm{M}(t_i)$ is far from the singularities of $\mathcal{R}$.

Let $\mathbf{k} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^\top$. Since the multi-camera trajectory has small pitch and roll, there are rotations $\mathtt{R_A}$ and $\mathtt{R_B}$ such that $\forall i, {\mathtt{R_A}}^{-1}\mathtt{R}_i^0{\mathtt{R_B}}^{-1}$ is almost a rotation around $\mathbf{k}$ (details in Appendix D.3.1). Let angles $(\alpha_i, \beta_i, \gamma_i)$ be such that $\mathcal{E}(\alpha_i, \beta_i, \gamma_i) = {\mathtt{R_A}}^{-1}\mathtt{R}_i^0{\mathtt{R_B}}^{-1}$ and $\mathcal{E}$ is defined in Eq.(6.3) and β_i is close to 0 (details in Appendix D.3.2). We initialize

$\mathbf{r}_{\mathrm{M}}(t_i) = \begin{pmatrix}\alpha_i & \beta_i & \gamma_i\end{pmatrix}^\top$. Thanks to Eqs.(6.4) and (6.3), we obtain

$$\mathcal{R}(\mathbf{r}_{\mathrm{M}}(t_i)) = \mathcal{R}(\alpha_i, \beta_i, \gamma_i) = \mathtt{R_A}\mathcal{E}(\alpha_i, \beta_i, \gamma_i)\mathtt{R_B} = \mathtt{R}_i^0. \tag{D.6}$$

According to Subsection 6.2.3 and Appendix D.2, $\mathbf{r}_{\mathrm{M}}(t_i)$ is a singularity of $\mathcal{E}$ iff $\beta_i \in \pi/2 + \pi\mathbb{Z}$. Since $\mathcal{R}$ has the same singularities as $\mathcal{E}$ (proof in Appendix D.3.3) and β_i is close to 0, $\mathbf{r}_{\mathrm{M}}(t_i)$ is far from the singularities of $\mathcal{R}$.

D.3.1 Technical Details: Estimate $\mathtt{R_A}$ and $\mathtt{R_B}$

Let $R(\mathbf{v}, \theta)$ be the rotation with axis $\mathbf{v}$ and angle θ. Since the multi-camera trajectory has small pitch and roll (Subsection 6.2.3), all $(\mathtt{R}_i^0)^\top\mathtt{R}_j^0$ are roughly rotations sharing a same axis $\mathbf{v} \in \mathbb{R}^3$. Thus there are rotation $\mathtt{R}$ and angle θ_i such that $\mathtt{R}_i^0 \approx \mathtt{R}R(\mathbf{v}, \theta_i)$. For all i and j,

$$(\mathtt{R}_i^0)^\top\mathtt{R}_j^0 \approx R(\mathbf{v}, \theta_j - \theta_i). \tag{D.7}$$

Let $\mathbf{v}_{i,j}$ be the axis of $(\mathtt{R}_i^0)^\top\mathtt{R}_j^0$.

First, we search $\mathbf{v}$ as the most colinear vector to all $\mathbf{v}_{i,j}$, i.e. $\mathbf{v}$ maximizes $\sum_{i,j}(\mathbf{v}_{i,j}^\top\mathbf{v})^2$. Thus $\mathbf{v}$ is the eigenvector of the largest eigenvalue of the symmetric matrix

$$\mathtt{C}_{\mathbf{v}} = \sum_{i,j}\mathbf{v}_{i,j}\mathbf{v}_{i,j}^\top. \tag{D.8}$$

Second, we estimate rotation $\tilde{\mathtt{R}}$ such that

$$\tilde{\mathtt{R}}\mathtt{R}_i^0 \approx R(\mathbf{v}, \theta_i') \tag{D.9}$$

Since $\mathtt{R}_i^0 \approx \mathtt{R}R(\mathbf{v}, \theta_i), \mathtt{R}_i^0\mathbf{v} \approx \mathtt{R}\mathbf{v}$. Let $\tilde{\mathbf{v}} = \sum_i \mathtt{R}_i^0\mathbf{v}/||\sum_i \mathtt{R}_i^0\mathbf{v}||$. Thus $\tilde{\mathbf{v}} \approx \mathtt{R}\mathbf{v} \approx \mathtt{R}_i^0\mathbf{v}$. Let $\tilde{\mathtt{R}}$ be a rotation such that $\tilde{\mathtt{R}}\tilde{\mathbf{v}} = \mathbf{v}$. Since $\tilde{\mathtt{R}}\mathtt{R}_i^0\mathbf{v} \approx \tilde{\mathtt{R}}\tilde{\mathbf{v}} = \mathbf{v}$, $\tilde{\mathtt{R}}\mathtt{R}_i^0 \approx R(\mathbf{v}, \theta_i')$.

Third, we estimate $\mathtt{R_A}$ and $\mathtt{R_B}$. Let $\mathtt{R}'$ be a rotation such that

$$\mathtt{R}'\mathbf{v} = \mathbf{k}. \tag{D.10}$$

We obtain $\mathtt{R}'\tilde{\mathtt{R}}\mathtt{R}_i^0\mathtt{R}'^\top \approx R(\mathbf{k}, \gamma_i)$. Thus

$$\mathtt{R_A}^{-1} = \mathtt{R}'\tilde{\mathtt{R}} \text{ and } \mathtt{R_B}^{-1} = \mathtt{R}'^\top. \tag{D.11}$$

D.3.2 Estimate $(\alpha_i, \beta_i, \gamma_i)$

Since $\mathcal{E}$ is surjective on the set of 3D rotations $\mathbb{SO}(3)$, the angles α_i, β_i and γ_i exist. Furthermore, they are defined up 2π multiples. We choose β_i that has the smallest $|\beta_i|$. Since $\mathcal{E}(\alpha_i, \beta_i, \gamma_i) \approx R(\mathbf{k}, \gamma_i) = \mathtt{R}_z(\gamma_i)$, $|\beta_i|$ is close to 0. We also remind that $\mathbf{r}_{\mathrm{M}}(t)$ is continuous (Subsection 6.2.1) and $|t_i - t_{i+1}|$ is small thanks to the keyframe sampling. Thus the γ_i series is chosen such that $|\gamma_i - \gamma_{i-1}|$ is as small as possible, and we do similarly for α_i ($|\beta_i - \beta_{i-1}|$ is also small) (see Subsection 6.3.3).

D.3.3 Proof: $\mathcal{R}$ and $\mathcal{E}$ have the same singularities

Here we show that $\ker \partial\mathcal{E} = \ker \partial(\mathtt{R_A}\mathcal{E}\mathtt{R_B})$ if $\mathtt{R_A}$ and $\mathtt{R_B}$ are two invertible 3×3 matrices. Let $\mathbf{x} \in \ker \partial\mathcal{E}$, $\mathcal{E}_{ij}$ be the coefficients of $\mathcal{E}$ and $\partial\mathcal{E}_{ij}$ be the gradient of $\mathcal{E}_{ij}$ with respect to parameters (α, β, γ). Thus $\partial\mathcal{E}_{ij}.\mathbf{x} = 0$ and

$$(\partial(\mathtt{R_A}\mathcal{E}\mathtt{R_B})_{ij}).\mathbf{x} = \left(\partial\left(\sum_{k,l} \mathtt{R_A}_{ik}\mathcal{E}_{kl}\mathtt{R_B}_{lj}\right)\right).\mathbf{x} = \sum_{k,l} \mathtt{R_A}_{ik}\mathtt{R_B}_{lj}(\partial\mathcal{E}_{kl}).\mathbf{x} = 0. \tag{D.12}$$

We see that $\partial(\mathtt{R_A}\mathcal{E}\mathtt{R_B}).\mathbf{x} = 0$, i.e. $\ker \partial\mathcal{E} \subseteq \ker \partial(\mathtt{R_A}\mathcal{E}\mathtt{R_B})$. Since $\mathtt{R_A}$ and $\mathtt{R_B}$ are invertible, we use this inclusion (replace $\mathcal{E}$ by $\mathtt{R_A}\mathcal{E}\mathtt{R_B}$, replace $\mathtt{R_A}$ by ${\mathtt{R_A}}^{-1}$, replace $\mathtt{R_B}$ by ${\mathtt{R_B}}^{-1}$) and obtain

$$\ker \partial(\mathtt{R_A}\mathcal{E}\mathtt{R_B}) \subseteq \ker \partial\left({\mathtt{R_A}}^{-1}(\mathtt{R_A}\mathcal{E}\mathtt{R_B}){\mathtt{R_B}}^{-1}\right) = \ker \partial\mathcal{E}. \tag{D.13}$$

Appendix E

Quaternion parametrization

E.1 A quick walk-through

Quaternion is an extension of complex numbers. The algebra of quaternions is equipped with addition and multiplication which is non commutative. (Some proofs will be ignored, curious readers can find out more details in [Dam+98]). The set of quaternions is equal to $\mathbb{R}^4$ and usually is denoted $\mathbb{H}$. We use the following forms:

Definition E.1. Let $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1, \mathbf{ij} = \mathbf{k}$ and $\mathbf{ji} = -\mathbf{k}$. Then $\mathbf{q} \in \mathbb{H}$ can be written:

$$\mathbf{q} \equiv [q_r, \mathbf{v}] \equiv [q_r, q_i, q_j, q_k] \equiv q_r + \mathbf{i}q_i + \mathbf{j}q_j + \mathbf{k}q_k$$

where $q_r, q_i, q_j, q_k \in \mathbb{R}$ and $\mathbf{v} \in \mathbb{R}^3$.

Quaternion addition and multiplication are a straightforward generalization of the addition and multiplication (respectively) of complex number using all four basic element $(1, i, j, k)$. For any two quaternions $\mathbf{q} = [q_r, \mathbf{v}]$ and $\mathbf{q}' = [q'_r, \mathbf{v}']$, the corresponding sum and product are given as follows:

$$\mathbf{q} + \mathbf{q}' = [q_r + q'_r, \mathbf{v} + \mathbf{v}'] \tag{E.1}$$

$$\mathbf{q}\mathbf{q}' = [q_r q'_r - \mathbf{v} \cdot \mathbf{v}', \mathbf{v} \times \mathbf{v}' + q_r \mathbf{v}' + q'_r \mathbf{v}] \tag{E.2}$$

$$r\mathbf{q} = \mathbf{q}r = [r, \mathbf{0}][q_r, \mathbf{v}] = [rq_r, r\mathbf{v}] \tag{E.3}$$

where $\cdot$ and $\times$ denote the dot and cross product in $\mathbb{R}^3$, respectively and $r \in \mathbb{R}$.

Corresponding to the definition of the conjugate of a complex number, we define the conjugate of a quaternion $\mathbf{q} = [q_r, \mathbf{v}]$:

$$\mathbf{q}^* \equiv [q_r, \mathbf{v}]^* \equiv [q_r, -\mathbf{v}]. \tag{E.4}$$

The norm of a quaternion $\mathbf{q} = [q_r, \mathbf{v}]$ is obtained using conjugation:

$$||\mathbf{q}|| \equiv \sqrt{\mathbf{q}\mathbf{q}^*} = \sqrt{q_r^2 + \mathbf{v} \cdot \mathbf{v}} = \sqrt{q_r^2 + q_i^2 + q_j^2 + q_k^2}. \tag{E.5}$$

Proposition E.2. *Let* $\mathbf{q}, \mathbf{p} \in \mathbb{H}$. *Then* $||\mathbf{qp}|| = ||\mathbf{q}||||\mathbf{p}||$.

Let $\mathbf{q}, \mathbf{q}' \in \mathbb{H}$, $\mathbf{q} = [q_r, \mathbf{v}] = [q_r, q_i, q_j, q_k]$, $\mathbf{q}' = [q'_r, \mathbf{v}'] = [q'_r, q'_i, q'_j, q'_k]$. The inner product is defined as $\bullet : \mathbb{H} \times \mathbb{H} \to \mathbb{R}$ where

$$\mathbf{q} \bullet \mathbf{q}' = \mathbf{q}' \bullet \mathbf{q} = q_r q'_r + \mathbf{v} \cdot \mathbf{v} = q_r q'_r + q_i q'_i + q_j q'_j + q_k q'_k. \tag{E.6}$$

Let α be the angle between them. Then $\mathbf{q} \bullet \mathbf{q}' = ||\mathbf{q}||||\mathbf{q}'|| \cos \alpha$.

We consider the set of quaternions $\breve{\mathbb{H}} = \mathbb{H} \setminus \{[0, 0, 0, 0]\}$. The element $I = [1, \mathbf{0}] \in \breve{\mathbb{H}}$ is the unique neutral element under quaternion multiplication.

Lemma E.3. *Let* $\mathbf{q} \in \breve{\mathbb{H}}$. *There exists* $\mathbf{q}^{-1} \in \mathbb{H}$ *such that* $\mathbf{q}\mathbf{q}^{-1} = \mathbf{q}^{-1} * \mathbf{q}$. *Furthermore,* $\mathbf{q}^{-1}$ *is unique and given by* $\mathbf{q}^{-1} = \dfrac{\mathbf{q}^*}{||\mathbf{q}||^2}$.

Now we focus on unit quaternion $\mathbf{q}$ which meets $||\mathbf{q}|| = 1$. We will use $\mathbb{H}_1$ to denote the set of unit quaternions. The set of unit quaternions constitutes a unit sphere in four-dimensional space and plays an important part in relation to general rotations.

Proposition E.4. *Let* $\mathbf{q} = [q_r, \mathbf{v}] \in \mathbb{H}_1$. *Then there exists* $\mathbf{v}' \in \mathbb{R}^3$ *and* $\theta \in]-\pi, \pi]$ *such that* $\mathbf{q} = [\cos\theta, \mathbf{v}' \sin\theta]$.

Quaternions perform rotation that is shown in the following propositions.

Proposition E.5. *Let* $\mathbf{p} \in \mathbb{H}$, $\mathbf{q} \in \breve{\mathbb{H}}$. *If* $r \in \mathbb{R} \setminus \{0\}$ *then* $(r\mathbf{q})\mathbf{p}(r\mathbf{q})^{-1} = \mathbf{q}\mathbf{p}\mathbf{q}^{-1}$.

Proposition E.6. *Let* $\mathbf{q} \in \mathbb{H}_1, \mathbf{q} = [\cos\theta, \sin\theta\mathbf{n}], ||\mathbf{n}|| = 1$. *Let* $\mathbf{r} = (x, y, z) \in \mathbb{R}^3$ *and* $\mathbf{p} = [0, \mathbf{r}] \in \mathbb{H}$. *Then* $\mathbf{p}' = \mathbf{q}\mathbf{p}\mathbf{q}^{-1}$ *is* $\mathbf{p}$ *rotated* 2θ *about the axis* $\mathbf{n}$.

In other words, given a unit vector $||\mathbf{n}||$ and a rotation angle θ, the unit quaternion $\mathbf{q} = [\cos\theta, \sin\theta\mathbf{n}]$ rotates $\mathbf{r} \in \mathbb{R}^3$ through the angle 2θ about $\mathbf{n}$. As a consequence of this proposition, any general three-dimensional rotation $\mathtt{R}$ about $\mathbf{n}$ ($||\mathbf{n}|| = 1$) with angle θ can be obtained by a unit quaternion. We compute rotation matrix $\mathtt{R}$ using quaternion $\mathbf{q} = [q_r, q_i, q_j, q_k]$ by

$$\mathtt{R} = \frac{\mathtt{S}}{||\mathbf{q}||^2} \tag{E.7}$$

where

$$\mathtt{S} = \begin{bmatrix} q_r^2 + q_i^2 - q_j^2 - q_k^2 & 2q_iq_j - 2q_rq_k & 2q_iq_k + 2q_rq_j \\ 2q_iq_j + 2q_rq_k & q_r^2 - q_i^2 + q_j^2 - q_k^2 & 2q_jq_k - 2q_rq_i \\ 2q_iq_k - 2q_rq_j & 2q_jq_k + 2q_rq_i & q_r^2 - q_i^2 - q_j^2 + q_k^2 \end{bmatrix}. \tag{E.8}$$

Rotation matrix to unit quaternion conversion.

Due to the fact that $||\mathbf{q}|| = q_r^2 + q_i^2 + q_j^2 + q_k^2 = 1$, it is straightforward to show that $\text{trace}(\mathtt{R}) = 4q_r^2 - 1$ where rotation $\mathtt{R}$ has a form as Eq.(D.2). So $q_r = \pm\frac{1}{2}\sqrt{1 + \text{trace}(\mathtt{R})}$ and

$$\mathbf{q} = \left[q_r,\ \frac{R_{32} - R_{23}}{4q_r},\ \frac{R_{13} - R_{31}}{4q_r},\ \frac{R_{21} - R_{12}}{4q_r}\right]. \tag{E.9}$$

This is numerically stable as long as the trace(R) is not close to -1; otherwise, we risk dividing by (nearly) zero. In that case, suppose R_{11} is the largest diagonal entry, so q_i will have the largest magnitude (the other cases are similar), then the following is safe

$$q_i = \pm\frac{1}{2}\sqrt{1 + R_{11} - R_{22} - R_{33}};\ \mathbf{q} = \left[\frac{R_{32} - R_{23}}{4q_i}, q_i,\ \frac{R_{12} + R_{21}}{4q_i},\ \frac{R_{13} + R_{31}}{4q_i}\right]. \tag{E.10}$$

E.2 The exponential and logarithm functions

We need quaternion versions of the real exponential and logarithm functions. The definitions and a few consequences of them are given here.

Let $\mathbf{q} \in \mathbb{H}_1$, where $\mathbf{q} = [\cos\theta, \sin\theta\mathbf{v}]$ as in Proposition E.4. The logarithm function log is defined

$$\log \mathbf{q} \equiv [0, \theta\mathbf{v}]. \tag{E.11}$$

Note that $\log[1,0,0,0] = [0,0,0,0]$ as in the real case. Note also that $\log \mathbf{q}$ is not in general a unit quaternion.

For a quaternion of the form $\mathbf{q} = [0, \theta\mathbf{v}],\ \theta \in \mathbb{R},\ \mathbf{v} \in \mathbb{R}^3,\ ||\mathbf{v}|| = 1$, the exponential function exp is defined by

$$\exp \mathbf{q} \equiv [\cos\theta, \sin\theta\mathbf{v}]. \tag{E.12}$$

Note that the exponential and logarithm functions are mutually inverse, and that exp maps into $\mathbb{H}_1$. From the above definition, we can define exponentiation for $\mathbf{q} \in \mathbb{H}_1$, $t \in \mathbb{R}$: Let $\mathbf{q} \in \mathbb{H}_1,\ t \in \mathbb{R}$. Exponentiation $\mathbf{q}^t$ is defined by

$$\mathbf{q}^t \equiv \exp(t \log \mathbf{q}). \tag{E.13}$$

This gives rise to the following:

Proposition E.7. *Let* $\mathbf{q} \in \mathbb{H}_1,\ t \in \mathbb{R}$. *Then* $\log(\mathbf{q}^t) = t \log \mathbf{q}$.

Proposition E.8. *Let* $\mathbf{q} \in \mathbb{H}_1$ *and* $a, b \in \mathbb{R}$. *Then* $\mathbf{q}^a\mathbf{q}^b = \mathbf{q}^{a+b}$.

Proposition E.9. *Let* $\mathbf{q} \in \mathbb{H}_1$ *and* $a, b \in \mathbb{R}$. *Then* $(\mathbf{q}^a)^b = \mathbf{q}^{ab}$.

E.3 Geometric intuition

We make some observations that can help the intuitive understanding of rotation with quaternion.

The unit quaternion q and $\mathbf{q}^{-1}$. Let $\mathbf{q} = [q_r, \mathbf{v}] \in \mathbb{H}_1$, then

$$[q_r, \mathbf{v}]^{-1} = \mathbf{q}^{-1} = \mathbf{q}^* = [q_r, -\mathbf{v}]. \tag{E.14}$$

$\mathbf{q}^{-1}$ rotates the same number of degrees as $\mathbf{q}$, but the axis points in the opposite direction. By inverting the axis and preserving the direction of rotation, a subsequent rotation by $\mathbf{q}^{-1}$ cancels out the effect of the rotation $\mathbf{q}$.

The quaternions q and $-\mathbf{q}$. From Eq.(E.7), $\mathtt{R}(\mathbf{q}) = \mathtt{R}(-\mathbf{q})$. The quaternion $-\mathbf{q}$ represents exactly the same rotation as $\mathbf{q}$. As a geometrical explanation, if $\mathbf{q} = [\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\mathbf{n}]$, then $-\mathbf{q} = [\cos\frac{\theta+2\pi}{2}, \sin\frac{\theta+2\pi}{2}\mathbf{n}]$. Rotations of angle θ and $\theta + 2\pi$ are the same.

Non-unit quaternions. Follows from Proposition E.5, all quaternions on the line $r\mathbf{q},\ r \in \mathbb{R},\ r \neq 0$ represent the same rotation.

Appendix F

Concepts on $\mathbb{SE}(3)$

F.1 Basic definitions

Throughout this appendix, we use the language of Lie groups and Lie algebras, but our development does not rely on anything other than elementary knowledge of the theory of Lie groups. A more in-deep treatment of some of the topics covered in this appendix can be found in [Gallier11], [Sattinger+13].

A d-dimensional *manifold* $\mathbb{M}$ is a topological space where every point $p \in \mathbb{M}$ is endowed with local Euclidean structure. Another way of saying is: the neighborhood of every point p is homeomorphic to an open ball in $\mathbb{R}^d$. (A function that maps from $\mathbb{M}$ to $\mathbb{R}^d$ is homeomorphic if it is a bicontinuous function, that is, both $f(\cdot)$ and its inverse $f(\cdot)^{-1}$ are continuous). For example, in the case of $\mathbb{M} = \mathbb{SO}(3)$, we have $d = 3$, so $\mathbb{SO}(3)$ is a 3-manifold.

A d-dimensional differentiable manifold $\mathbb{M}$ embedded in $\mathbb{R}^n$ (with $n \geq d$) has an associated d-dimensional *tangent space* for every point $p \in \mathbb{M}$. This space is denoted as $\mathbb{T}_p\mathbb{M}$. The points p are called non-singular points if the dimension of the tangent space at p is identical to that of the manifold; otherwise, they are called singular points and a curve that crossed itself does not have a unique tangent line at these points. Informally, a tangent space can be visualized as the vector space of the derivatives at p of all possible smooth curves that pass through p, e.g. $\mathbb{T}_p\mathbb{M}$ contains all the possible "velocity" vectors of a particle at p and constrained to $\mathbb{M}$.

Geodesics are defined simply to be locally shortest path on a manifold $\mathbb{M}$. In the case of $\mathbb{M} = \mathbb{R}^d$, geodesics are straight lines.

A *Lie group* is a group G which is at the same time a differentiable manifold having the property that a mapping $G \to G$ induced by left or right multiplication by a fixed element $g \in G$ is smooth and the mapping $g \mapsto g^{-1}$ is smooth. For example, $\mathbb{SO}(3)$ and $\mathbb{SE}(3)$ are Lie groups. By the way, $\mathbb{R}^n$ can be also considered a Lie group for any $n \geq 1$.

Associated with a Lie group G, there is a *Lie algebra* $\mathfrak{g}$. The connection between these two entities is the *exponential map* taking an element $\Omega \in \mathfrak{g}$ to its (matrix) exponential $\exp(\Omega)$ which is an element in Lie group G.

F.2 $\mathbb{SE}(3)$ as a Lie group

F.2.1 Properties

The special Euclidean group of transformation is denoted as $\mathbb{SE}(3)$, and its members are the set of 4×4 matrices with this structure:

$$\mathtt{T} = \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{F.1}$$

with $\mathtt{R} \in \mathbb{SO}(3)$, $\mathbf{t} = [t_x, t_y, t_z]^\top \in \mathbb{R}^3$ and group product the standard matrix product. $\mathbb{SE}(3)$ is a 6-dimensional manifold, i.e. it has 6 DoF. Three DoF correspond to the 3D translation vector and the other three DoF to the rotation. $\mathbb{SE}(3)$ is *not* isomorphic to $\mathbb{SO}(3) \times \mathbb{R}^3$ as a group, since the group multiplication of both groups are different. It is said that $\mathbb{SE}(3)$ is a semidirect product of the groups $\mathbb{SO}(3)$ and $\mathbb{R}^3$.

F.2.2 Lie algebra of $\mathbb{SO}(3)$

Since $\mathbb{SE}(3)$ has the manifold structure of the product $\mathbb{SO}(3) \times \mathbb{R}^3$, it makes sense to define first the properties of $\mathbb{SO}(3)$, which is also a Lie group. The group $\mathbb{SO}(3)$ has an associated Lie algebra $\mathfrak{so}(3)$ consisting of the set of all skew-symmetric 3×3 matrices. The generators of $\mathfrak{so}(3)$ are

$$\begin{aligned} \mathfrak{so}(3) &= \left\{ \mathtt{G}_i^{\mathfrak{so}(3)} \right\}_{i=1,2,3}, \quad \mathtt{G}_1^{\mathfrak{so}(3)} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}_\times = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \\ \mathtt{G}_2^{\mathfrak{so}(3)} &= \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}_\times = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \quad \mathtt{G}_3^{\mathfrak{so}(3)} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}_\times = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{aligned} \tag{F.2}$$

where the skew-symmetric matrix operator $[\cdot]_\times$ is defined in Eq.(2.12).

The *exponential map* of $\mathfrak{so}(3)$ is

$$\begin{aligned} \exp : \mathfrak{so}(3) &\rightarrow \mathbb{SO}(3) \\ [\omega]_\times &\mapsto \mathtt{R}_{3\times 3} = \sum_{i \geq 0} \frac{([\omega]_\times)^i}{i!}. \end{aligned} \tag{F.3}$$

It corresponds to the matrix exponentiation. The closed-form solution can be computed using the Rodrigues' formula:

$$e^{\omega} \equiv e^{[\omega]_\times} = \begin{cases} \mathtt{I}_3 + [\omega]_\times + \dfrac{1}{2}[\omega]_\times^2 & \text{for } \theta \to 0 \\ \mathtt{I}_3 + \dfrac{\sin\theta}{\theta}[\omega]_\times + \dfrac{1-\cos\theta}{\theta^2}[\omega]_\times^2 & \text{else} \end{cases} \tag{F.4}$$

where the angle $\theta = ||\omega||_2$ and $[\omega]_\times$ is the skew symmetric matrix generated by the 3-vector ω.

The *logarithm map* is

$$\begin{aligned} \log : \mathbb{SO}(3) &\rightarrow \mathfrak{so}(3) \\ \mathtt{R}_{3\times 3} &\mapsto [\omega]_\times. \end{aligned} \tag{F.5}$$

It is the inverse of the exp function defined above and corresponds to the logarithm of the 3×3 rotation matrices. The Rodrigues rotation formula gives:

$$\log(\mathtt{R}) = \begin{cases} \frac{1}{2}(\mathtt{R} - \mathtt{R}^\top) = \mathtt{0}_{3\times 3} & \text{for } d \to \pm 1 \\ \frac{\arccos(d)}{2\sqrt{1-d^2}}(\mathtt{R} - \mathtt{R}^\top) & \text{for } d \in (-1, 1) \end{cases} \quad \text{where } d = \frac{1}{2}(\text{trace}(\mathtt{R}) - 1). \tag{F.6}$$

F.2.3 Lie algebra of $\mathbb{SE}(3)$

The group $\mathbb{SE}(3)$ has an associated Lie algebra $\mathfrak{se}(3)$ whose generators are six 4×4 matrices, each corresponding to either rotation or translation along each axis:

$$\begin{aligned} \mathfrak{se}(3) = \left\{ \mathtt{G}_i^{\mathfrak{se}(3)} \right\}_{i=1,\ldots,6}, \mathtt{G}_{1,2,3}^{\mathfrak{se}(3)} = \left[\begin{array}{c|c} \mathtt{G}_{\{1,2,3\}}^{\mathfrak{so}(3)} & \begin{matrix}0\\0\\0\end{matrix} \\ \hline \mathbf{0}_{3\times 1} & 0 \end{array}\right], \\ \mathtt{G}_4^{\mathfrak{se}(3)} = \left[\begin{array}{c|c} \mathtt{0}_{3\times 3} & \begin{matrix}1\\0\\0\end{matrix} \\ \hline \mathbf{0}_{3\times 1} & 0 \end{array}\right], \mathtt{G}_5^{\mathfrak{se}(3)} = \left[\begin{array}{c|c} \mathtt{0}_{3\times 3} & \begin{matrix}0\\1\\0\end{matrix} \\ \hline \mathbf{0}_{3\times 1} & 0 \end{array}\right], \mathtt{G}_6^{\mathfrak{se}(3)} = \left[\begin{array}{c|c} \mathtt{0}_{3\times 3} & \begin{matrix}0\\0\\1\end{matrix} \\ \hline \mathbf{0}_{3\times 1} & 0 \end{array}\right]. \end{aligned} \tag{F.7}$$

The *exponential* map

$$\exp : \mathfrak{se}(3) \to \mathbb{SE}(3) \tag{F.8}$$

has the closed form:

$$\exp(\upsilon, \omega)_{\mathfrak{se}(3)} = \begin{bmatrix} \exp([\omega]_\times) & \mathtt{V}\upsilon \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{SE}(3) \tag{F.9}$$

where the linear map $\mathtt{V} : \mathbb{R}^3 \to \mathbb{R}^3$ is invertible and has the closed-form solution [Gallier11]:

$$\mathtt{V} = \begin{cases} \mathtt{I} + \frac{1}{2}[\omega]_\times + \frac{1}{6}[\omega]_\times^2 & \text{for } \theta \to 0 \\ \mathtt{I} + \frac{1-\cos\theta}{\theta^2}[\omega]_\times + \frac{\theta - \sin\theta}{\theta^3}[\omega]_\times^2 & \text{else} \end{cases} \tag{F.10}$$

where the angle $\theta = ||\omega||_2$.

The *logarithm map* is

$$\begin{array}{rcl} \log : \mathbb{SE}(3) & \to & \mathfrak{se}(3) \\ \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} & \mapsto & (\upsilon, \omega) \end{array} \tag{F.11}$$

where $[\omega]_\times = \log(\mathtt{R})$ (see Eq.(F.6)) and $\upsilon = \mathtt{V}^{-1}\mathbf{t}$ with $\mathtt{V}$ defined in Eq.(F.10).

Appendix G

Rolling shutter calibration using strobe

Stroboscope STA10K (langlois-france.com) with xenon lamp and a power of 40W (Figure G.1) is used in our setup. The rotation speed v_r in revolutions/minute (rpm) is indicated by LED digital display. The period T_o of stroboscope flash is calculated by $60/v_r$ in seconds. As can be seen in Figure G.2 and G.3, the images exposed by strobe flash have two or more distinct parts corresponding to on and off periods of the strobe flash due to the sequential read-out on the CMOS chip. One part of the image is brightly lit by a flash light, while the next part is dark and unlit, as the flash was off by the the time that the part of the CMOS was sequenced. The difference between two parts of the same frame can be observed in Figure G.2 for Gopro camera and the top corners in Figure G.3 for Ricoh Theta S camera.

If we measure the number of image rows N during a flash period, the line delay can be obtained as:

$$\tau = \frac{T_o}{N} = \frac{60}{v_r N} \tag{G.1}$$

where v_r is rotation speed indicated by LED digital display. For the first case Gopro in Figure G.2, $v_r = 8442$ rpm, $N = 781$ (315 lit lines + 466 unlit lines) on average, so $\tau = 9.10$ μs. For the second case Ricoh Theta S in Figure G.3, $v_r = 8980$ rpm, $N = 208$ (108 lit lines + 100 unlit lines) on average, we obtain $\tau = 32.1$ μs. Furthermore, in Figure G.3, the sequences of bands of the flash light in left and right images are synchronous visibly. We can verify that two cameras are synchronous.

FIGURE G.1: Stroboscope STA10K.

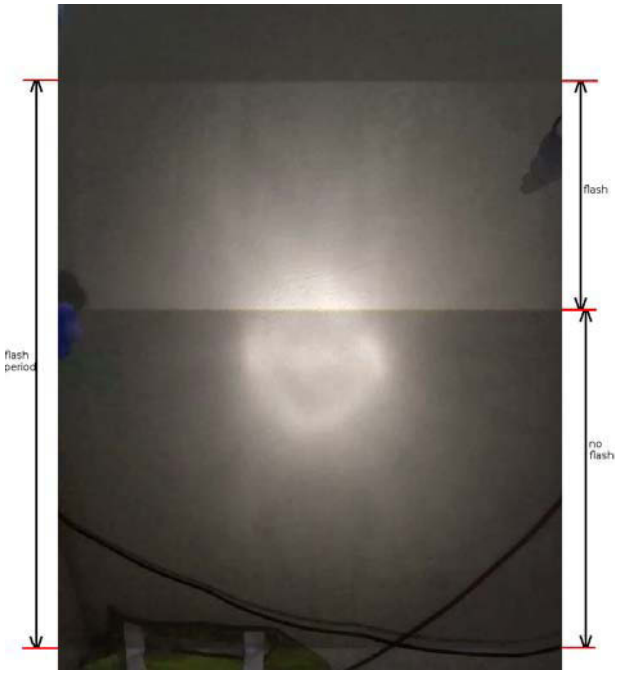


FIGURE G.2: Gopro camera observing a wall periodically flashed by a stroboscope (best viewed in the electronic version).

FIGURE G.3: Ricoh Theta S camera observing a room periodically flashed by a stroboscope. The flash/no flash can be seen on the wall at the top of the circular images (best viewed in the electronic version).

Appendix H

Sparsity of $\mathtt{Z}$ in the RCS

H.1 Notations and prerequisites

If L_1 and L_2 are two lists of integers, we use notations

$$L_1 + L_2 = \{i_1 + i_2, i_1 \in L_1, i_2 \in L_2\}, \tag{H.1}$$

$$L_1 \times L_2 = \{(i_1, i_2), i_1 \in L_1, i_2 \in L_2\}, \tag{H.2}$$

$$(L_1)^2 = L_1 \times L_1. \tag{H.3}$$

We also use Eq.(H.1) if L_1 and L_2 are two lists of integer pairs. If a matrix $\mathtt{A}$ is partitioned by blocks $\mathtt{A}_i$ (horizontally or vertically), we define an index list

$$L(\mathtt{A}) = \{i, \mathtt{A}_i \neq 0\}. \tag{H.4}$$

If a matrix $\mathtt{A}$ is partitioned by blocks $\mathtt{A}_{ij}$ (both horizontally and vertically), we define a list of index pairs

$$L(\mathtt{A}) = \{(i,j), \mathtt{A}_{i,j} \neq 0\}. \tag{H.5}$$

If matrices $\mathtt{A}$ and $\mathtt{B}$ are horizontally partitioned by blocks $\mathtt{A}_i$ and $\mathtt{B}_j$ respectively, $\mathtt{A}^\top\mathtt{B}$ is partitioned by blocks $\mathtt{A}_i^\top\mathtt{B}_j$ and for these blocks, we have

$$L(\mathtt{A}^\top\mathtt{B}) \subseteq L(\mathtt{A}) \times L(\mathtt{B}). \tag{H.6}$$

If $\mathtt{C} = \sum_i \mathtt{C}_i$ and all $\mathtt{C}_i$ have the same block partition,

$$L(\mathtt{C}) \subseteq \cup_i L(\mathtt{C}_i). \tag{H.7}$$

If all blocks considered by L have the same size $a \times b$, we can write $L_{a\times b}$ instead of L.

In practice, it is improbable that a product or sum of non-zero matrices is zero. Thus, we replace the inclusions above by equalities in our proof, i.e. we use

$$L(\mathtt{A}^\top\mathtt{B}) = L(\mathtt{A}) \times L(\mathtt{B}) \text{ and } L(\mathtt{C}) = \cup_i L(\mathtt{C}_i). \tag{H.8}$$

Let $\mathtt{Z}^g$ and $\mathtt{Z}^r$ be the $6(m+1) \times 6(m+1)$ top-left blocks of the RCS in the standard case (GS in Chapter 5) and the (RS, SFA) case in Chapter 6 (more details in

Section 6.5.2), respectively. In the next section, we show that

$$L_{6\times 6}(\mathtt{Z}^r) = L_{6\times 6}(\mathtt{Z}^g) + \{-1,0,+1\}^2. \tag{H.9}$$

In these expressions and the following ones, we implicitly omit integers that are below 0 and above m (e.g. we omit $i-1$ if $i=0$ and omit $i+1$ if $i=m$).

H.2 Proof of Eq.(H.9)

The image projection function of the l-th 3D point in the i-th multi-camera pose is $\varphi^g_{i,l}$ in the standard GS case (in Chapter 5) and $\varphi^r_{i,l}$ in the (RS, SFA) case in Chapter 6. According to Section 6.5.2, we have

$$\varphi^g_{i,l} = \varphi(\mathbf{m}_i, \mathbf{m}', \mathbf{X}^l) \text{ and } \varphi^r_{i,l} = \varphi(\mathbf{m}_{i-1}, \mathbf{m}_i, \mathbf{m}_{i+1}, \mathbf{m}', \mathbf{X}^l). \tag{H.10}$$

Thus, $\frac{\partial \varphi^g_{i,l}}{\partial \mathbf{m}_{i'}} \neq \mathtt{0}$ iff $i'=i$, and $\frac{\partial \varphi^r_{i,l}}{\partial \mathbf{m}_{i'}} \neq \mathtt{0}$ iff $i' \in \{i-1,i,i+1\}$. We rewrite this using the notations in Appendix H.1:

$$L_{2\times 6}\left(\frac{\partial \varphi^g_{i,l}}{\partial \mathbf{M}}\right) = \{i\} \text{ and } L_{2\times 6}\left(\frac{\partial \varphi^r_{i,l}}{\partial \mathbf{M}}\right) = \{i-1,i,i+1\}. \tag{H.11}$$

Notations $\varphi_{i,l}$ and $\mathtt{Z}, \mathtt{U}, \ldots$ are used in expressions that hold for both "r" and "g" upper-indices added to these notations. Let V_l be the list of keyframe indices where the l-th point is inlier. According to Eqs.(5.12), (5.16), (6.46) and (6.47), we have $\mathtt{Z} = \mathtt{U} - \mathtt{W}\mathtt{V}^{-1}\mathtt{W}^\top$ where

$$\mathtt{U} = \sum_{i\in V_l} \left(\frac{\partial \varphi_{i,l}}{\partial \mathbf{M}}\right)^\top \frac{\partial \varphi_{i,l}}{\partial \mathbf{M}}, \mathtt{W} = \sum_{i\in V_l} \left(\frac{\partial \varphi_{i,l}}{\partial \mathbf{M}}\right)^\top \frac{\partial \varphi_{i,l}}{\partial \mathbf{X}}, \mathtt{V} = \sum_{i\in V_l} \left(\frac{\partial \varphi_{i,l}}{\partial \mathbf{X}}\right)^\top \frac{\partial \varphi_{i,l}}{\partial \mathbf{X}}. \tag{H.12}$$

Sine $L_{6\times 6}(\mathtt{U}) = \cup_{i\in V_l}\left(L_{2\times 6}\left(\frac{\partial \varphi_{i,l}}{\partial \mathbf{M}}\right)\right)^2$, we have

$$\begin{array}{rcl} L_{6\times 6}(\mathtt{U}^g) & = & \cup_i\{(i,i)\} \text{ and} \\ L_{6\times 6}(\mathtt{U}^r) & = & \cup_i\{(i-1,i,i+1)\}^2 = \{(-1,0,+1)\}^2 + \cup_i\{(i,i)\}. \end{array} \tag{H.13}$$

Furthermore, $\mathtt{W}$ in Eqs.(5.12) and (6.46) are horizontally partitioned in blocks $\mathtt{W}_l = \sum_{i\in V_l}\left(\frac{\partial \varphi_{i,l}}{\partial \mathbf{M}}\right)^\top \frac{\partial \varphi_{i,l}}{\partial \mathbf{X}^l}$. Since

$$L_{6\times 3}(\mathtt{W}_l) = \cup_{i\in V_l} L_{6\times 3}\left(\left(\frac{\partial \varphi_{i,l}}{\partial \mathbf{M}}\right)^\top \frac{\partial \varphi_{i,l}}{\partial \mathbf{X}^l}\right) = \cup_{i\in V_l} L_{2\times 6}\left(\frac{\partial \varphi_{i,l}}{\partial \mathbf{M}}\right), \tag{H.14}$$

we have

$$L_{6\times 3}(\mathtt{W}^g_l) = V_l \text{ and } L_{6\times 3}(\mathtt{W}^r_l) = V_l + \{-1,0,+1\}. \tag{H.15}$$

Since $\mathtt{V}$ in in Eqs.(5.12) and (6.46) are block-wise diagonal with invertible blocks $\mathtt{V}_l = \sum_{i\in V_l}\left(\frac{\partial \varphi_{i,l}}{\partial \mathbf{X}^l}\right)^\top \frac{\partial \varphi_{i,l}}{\partial \mathbf{X}^l}$ and $\mathtt{W}\mathtt{V}^{-1}\mathtt{W}^\top = \sum_l \mathtt{W}_l\mathtt{V}_l^{-1}\mathtt{W}_l^\top$, we have $L_{6\times 6}(\mathtt{W}\mathtt{V}^{-1}\mathtt{W}^\top) = \cup_l (L_{6\times 3}(\mathtt{W}_l))^2$ Thus,

$$\begin{array}{rcl} L_{6\times 6}(\mathtt{W}^g\mathtt{V}^{-1}(\mathtt{W}^g)^\top) & = & \cup_l (V_l)^2 \text{ and} \\ L_{6\times 6}(\mathtt{W}^r\mathtt{V}^{-1}(\mathtt{W}^r)^\top) & = & \cup_l (V_l + \{-1,0,+1\})^2 = \{-1,0,+1\}^2 + \cup_l (V_l)^2. \end{array} \tag{H.16}$$

Thus,

$$\begin{array}{rcl} L_{6\times 6}(\mathtt{Z}^g) & = & \cup_i (i,i) \cup_l (V_l)^2 \text{ and} \\ L_{6\times 6}(\mathtt{Z}^r) & = & \{-1,0,+1\}^2 + (\cup_i \{(i,i)\} \cup_l (V_l)^2). \end{array} \tag{H.17}$$

We obtain Eq.(H.9).

Annexe I

Résumé étendu en français

De nombreuses caméras à 360^o et sphériques existent maintenant sur le marché (par exemple, une multi-caméra composée de plusieurs caméras Gopro rigidement liées [360rize] ; ou [ThetaS], [Gear360], [Virb360], [Ladybug2]). Avant son utilisation, une telle caméra doit tout d'abord être synchronisée et étalonnée. Les travaux de cette thèse proposent des méthodes souples permettant de synchroniser et d'auto-étalonner (intrinsèquement et extrinsèquement) plusieurs caméras grand public sans faire l'hypothèse qu'elles ont des champs de vue communs, et sans utiliser de mire d'étalonnage ou de cibles.

Sommaire

I.1 Introduction

I.1.1 Motivation

Les caméras omnidirectionnelles sont utilisées, entre autres, dans des applications de surveillance, navigation, robotique, réalité virtuelle, télé-présence, etc. Certaines applications nécessitent un modèle numérique d'un environnement. A côte des scanners (LIDAR, temps de vol, lumière structure) qui sont très précis mais coûteux et difficilement embarquables, les caméras sont un choix économique et convivial pour l'utilisateur. De plus, la modélisation efficace d'un environnement complet nécessite une acquisition sur (quasiment) l'ensemble de la sphère de vue. C'est une des raisons pour lesquelles, on considère ici un capteur omnidirectionnel et "multi-caméra" formé de plusieurs caméras grand public fixées rigidement. Par exemple, nous utilisons quatre caméras GoPro [GoPro] enfermées dans une boîte en carton et fixées sur un casque (Figure I.1). L'utilisateur se déplace à pied, vélo ou autres dans des environnements urbains et naturels. On se concentre ici sur les étapes de synchronisation et d'auto-étalonnage de la multi-caméra.

La synchronisation est essentielle afin d'intégrer l'information disponible à partir de plusieurs vidéos simultanément. Même si une télécommande Wifi permet de démarrer toutes les vidéos en même temps en un clic, cette synchronisation n'est pas assez précise pour des applications de vidéo 360 ou modélisation 3D (l'écart entre deux vidéos est de l'ordre de 0.04 seconde et peut parfois dépasser 0.1 seconde) pour les GoPro Hero3. De plus, à chaque utilisation de la télécommande, les décalages temporels entre vidéos ne sont pas les mêmes.

L'étalonnage consiste à trouver la relation (définie par une fonction) entre un point 3D dans l'espace et sa projection dans l'image de la (multi-)caméra. On modélise un système multi-caméra par un ensemble de n caméras distinctes de la manière suivante : un modèle de caméra est choisi pour les *paramètres intrinsèques* de chaque caméra ; on définit pour chaque caméra une transformation dans le repère multi-caméra (ce sont les *paramètres extrinsèques*). Si besoin est, des matrices de passage entre deux caméras (leurs poses relatives) peuvent s'exprimer avec les paramètres extrinsèques. De plus, on initialise les paramètres intrinsèques grâce à des hypothèses appropriées à une caméra omnidirectionnelle sans direction privilégiée : toutes les caméras ont la même configuration (fréquence, résolution, champs de vue) et sont approximativement équiangulaires. On peut mettre beaucoup de caméras afin d'augmenter les recouvrements de champs de vues (ce qui facilite la mise en correspondance entre caméras), mais la baseline entre caméras et le prix augmentent. Dans notre cas, ces recouvrements sont trop petits pour obtenir automatiquement une mise en correspondance entre caméras. On développe une méthode flexible qui permet d'étalonner le système multi-caméra. La flexibilité est liée à la simplicité de la mise en oeuvre : pas besoin de mire, ni de connaissance précise de la géométrie.

Les caméras d'entrée de gamme sont "rolling shutter" (RS), et ceci est dû à la technologie CMOS ("Complementary Metal Oxide Semiconductor"). Cette technologie est très prisée pour les capteurs à faible coût en raison des nombreux avantages : faible consommation électrique ; simplicité du circuit et donc encombrement réduit ; et un coût très avantageux. CMOS est opposée à la technologie CCD ("Charge-Coupled Device") qui est généralement utilisée pour fabriquer les capteurs haut de gamme ou nécessitant un rapport signal sur bruit très élevé. Pour une caméra RS, chaque ligne de pixels est

acquise à un instant différent (autrement, une caméra est "global shutter" (GS) et tous ses pixels ont la même date). Lorsque la scène est figée (avec illumination constante) et que la caméra est fixée, l'image obtenue ne contient que les distorsions résultant du bruit de capteur et du système optique, comme c'est le cas avec une caméra GS. En revanche si la caméra se déplace ou que la scène est mobile ou que l'illumination change soudainement, alors des distorsions ou artefacts apparaissent dans l'image ou la vidéo acquise. Ceux-ci sont d'autant plus notables à l'oeil que la vitesse de l'utilisateur augmente.

I.1.2 Travaux antérieurs

Synchronisation. L'article [Gaspar+14] propose un état de l'art sur la synchronisation de vidéos, mais ces méthodes nécessitent une mise en correspondance entre caméras ou un champ de vue (partiellement) recouvrant, ou sont conçues pour un système de caméras non rigide. Certaines méthodes ont une précision "sous-trame" : le décalage temporel entre deux vidéos est un nombre d'images non entier (un réel). Plusieurs méthodes exploitent les points d'intérêt observés dans les vidéos et alignent des vidéos en se basant sur des caractéristiques visuelles. Une méthode voisine de la notre [Spencer+04] évite une mise en correspondance entre caméras : elle calcule pour chaque vidéo une suite de transformations entre images successives, et le décalage temporel est celui qui corrèle le mieux les suites de deux vidéos. Un exemple intuitif est celui où la transformation est la translation 2D calculée avec un suivi de points : plus la translation est grande dans une vidéo, plus la translation est grande dans une autre.

Auto-étalonnage de caméras. Dans le cas d'une caméra (monoculaire), il y a une approche classique [Hartley+04], [Triggs+00] pour estimer les paramètres intrinsèques. Ses étapes sont : reconstruction projective, auto-calibration (ex : en supposant que les pixels sont carrés), et ajustement de faisceaux (AF). La distorsion radiale peut aussi se calculer [Micusik+06]. Dans le cas d'une multi-caméra, il faut d'abord une estimation initiale des poses entre caméras (i.e. rotations et translations relatives entre caméras). On peut les calculer à partir de reconstructions obtenues séparément pour chaque caméra, et si elles sont synchronisées. Il y a deux types de méthodes pour cela : des récalages basés sur les points 3D [Carrera+11] ou sur les poses [Esquivel+07]. Dans le premier cas, les points sont mis en correspondance grâce à leur projections dans deux caméras différentes embarquées sur un robot (qui doit faire un tour sur lui-même) et par l'estimation robuste d'une similitude. Dans le second cas, la contrainte de rigidité entre les différents capteurs du système multi-caméra en mouvement est utilisée. L'évolution temporelle de la pose de chaque caméra est déterminée. Puis, on déduit la pose relative entre les différentes caméras (paramètres extrinsèques) qui ne varie pas au cours du temps. Le mouvement ne doit pas être une translation pure.

Il faut enfin raffiner la géométrie multi-caméra avec un AF. L'AF de [Lébraly+11] raffine les poses relatives entre caméras, en plus des poses multi-caméras et des points 3D, en minimisant une erreur de reprojection dans l'espace image rectifié du modèle de distorsion polynomial classique [Lavest+98], [Sturm+11]. L'AF de [Schneider+13] traite le cas de points à l'infini et utilise les directions de rayons comme observations. Le raffinement simultané avec les paramètres intrinsèques (focale, distorsion, etc) n'est suggéré qu'en travaux futurs dans [Lébraly+11] et [Schneider+13].

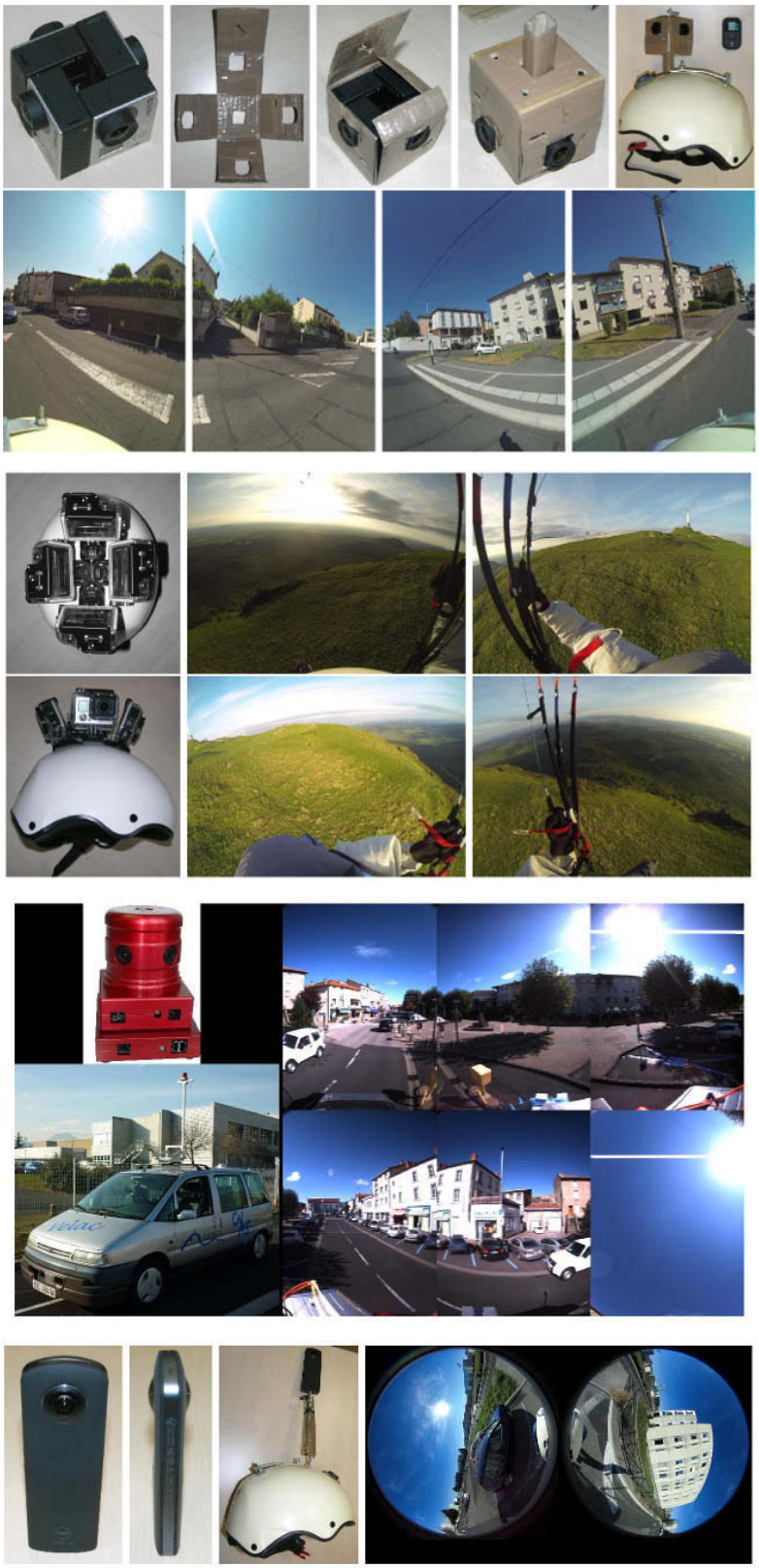


FIGURE I.1: Multi-caméras et images capturées. De haut en bas : quatre caméras Gopro Hero3 enfermées dans une boîte en carton ; quatre caméras Gopro Hero3 dans leurs coques ; PointGrey Ladybug2 et Ricoh Theta S.

Ajustement de faisceaux pour une caméra rolling shutter. Les AFs antérieurs monoculaires estiment le coefficient de RS ("line delay") en supposant que les points 3D sont connus et sur une mire d'étalonnage [Oth+13] ; ou imposent ce coefficient de RS connu [Hedborg+12], [Duchamp+15]. Les AFs précédents de multi-caméra n'estiment ni la synchronisation ni le coefficient de RS ; seuls [Klingner+13] et [Saurer+16] traitent le cas où le coefficient de RS est connu mais ont besoin d'autres capteurs.

Chaque AF RS a un modèle de la trajectoire de la caméra qui fournit la position de la caméra à chaque ligne d'une trame, et qui devrait avoir un nombre modéré de paramètres à estimer. Dans l'article [Hedborg+12], une pose est estimée à chaque image par AF et les poses entre deux trames consécutives sont interpolées à partir des poses de ces deux trames. L'AF dans [Duchamp+15] ajoute des paramètres supplémentaires pour éviter cette hypothèse d'interpolation linéaire : cette méthode optimise non seulement la pose mais aussi des vitesses de rotation et de translation à chaque image clé. Dans [Oth+13], un modèle de trajectoire en temps continu utilise des B-splines et l'AF optimise les noeuds des splines. Cette méthode choisit le nombre de noeuds et initialise leur distribution le long de la trajectoire. Dans [Klingner+13], la pose relative entre une pose inter-trame et une pose de trame optimisée est fournie par une centrale inertielle (IMU) à haute fréquence. Dans le travail de [Saurer+16], les vitesses de rotation et de translation sont également estimées à chaque image (il y a seulement 4 images par seconde) et l'AF applique une contrainte de pose relative en utilisant les données GPS/INS. Les méthodes [Hedborg+12], [Oth+13], [Duchamp+15] sont expérimentées sur des trajectoires de caméras de quelques mètres de long. Notre approche est purement visuelle et traite de trajectoires plus longues (centaines de mètres ou kilomètres) car elle estime seulement des poses aux images clés.

Dans le contexte d'un multi-capteur général, [Furgale+13] estime simultanément les décalages en temps et en espace entre les capteurs. Dans les expériences, le multi-capteur est composé d'une caméra et d'une IMU. La meilleure précision est obtenue grâce à l'utilisation de toutes les mesures à la fois, une représentation continue (une B-spline pour les poses IMU) et un estimateur au maximum de vraisemblance des paramètres (décalage temporel, transformation entre IMU et caméra, poses IMU et autres). [Lovegrove+13] propose un auto-étalonnage (synchronisation, poses relatives, paramètres intrinsèques) pour un multi-capteur inertiel-caméra par un AF local. Grâce à un paramétrage adéquat du mouvement continu, il traite également des caméras RS et a un meilleur paramétrage des rotations. En effet, il évite les singularités du paramétrage global et minimal des rotations (qu'il y a dans [Furgale+13]), mais suppose que le temps entre deux images clés consécutives est constant (uniforme). Nous proposons un paramétrage minimal global de la rotation et traitons la distribution non-uniforme des images clés fournies par le Structure-from-Motion (SfM) standard [Mouragnon+07], [Mouragnon+09].

I.1.3 Contributions

Nos contributions ne portent pas sur l'initialisation des paramètres intrinsèques/extrinsèques mais sur la synchronisation, l'AF et le rolling shutter. Ici, on applique des SfM monoculaire et multi-caméras [Mouragnon+07], [Mouragnon+09] avec des paramètres intrinsèques approximativement connus, qui sont ensuite raffinés par AF. On a une estimation initiale des paramètres intrinsèques de chaque caméra car on connaît approximativement leur champs de vue et elles sont approximativement équiangulaires

(Section I.1.1). On a aussi une estimation initiale des poses relatives entre caméras (mêmes centres, rotation d'angle $2\pi/n$ entre deux caméras adjacentes pour n caméras).

Synchronisation. On n'utilise pas directement les points d'intérêt mais les poses successives de chaque caméra calculées par SfM monoculaire. Comme les caméras sont rigidement liées les unes aux autres, elles ont la même vitesse de rotation au même instant. On calcule donc une vitesse de rotation au cours du temps pour chaque caméra, et on cherche les décalages qui les font le mieux correspondre. La transformation utilisée dans [Spencer+04] n'est pas une rotation mais une translation (heuristique) ou homographie/matrice fondamentale qui ne tient pas compte de la distorsion radiale des caméras.

Ajustement de faisceaux. Nous améliorons de deux façons l'AF dans [Lébraly+11] qui est basé sur le modèle de distorsion polynomial classique [Lavest+98], [Sturm+11] pour une caméra. Notre AF raffine les paramètres intrinsèques, pas seulement les paramètres extrinsèques entre caméras et les autres paramètres 3D (poses multi-caméras et points 3D). Il minimise l'erreur de reprojection dans l'espace image original (i.e. l'espace "distordu" où les points d'images sont détectés), pas dans l'espace image rectifié. Sous l'hypothèse habituelle que les bruits d'images sont de moyennes nulles, indépendants, et ont des distributions normales et identiques, notre AF est un estimateur au maximum de vraisemblance (cette hypothèse n'est pas vraie dans l'espace image rectifié).

Nous proposons une nouvelle méthode flexible pour obtenir non seulement les paramètres habituels mais aussi la synchronisation à précision sous-trame et le coefficient de rolling shutter. On commence par une calibration initiale avec le modèle d'une caméra le plus simple (le modèle de caméra GS) et la synchronisation à précision trame (FA) en utilisant les méthodes précédants. Notre AF fournit la synchronisation à précision sous-trame (SFA), i.e. il estime les décalages à précision réelle entre une vidéo de référence et les autres. Il estime également le coefficient de RS, i.e. le décalage séparant l'acquisition de deux lignes consécutives de l'image. De plus, on fait des expériences sur des trajectoires longues (centaines de mètres ou kilomètres) sans capteur additionnel.

I.2 Vue générale de notre méthode

D'abord, le modèle de caméra monoculaire est initialisé comme indiqué dans la Section I.3 en supposant que les fisheyes sont approximativement équiangulaires et en utilisant une connaissance approximative de leur angle de champs de vue. Les modèles de caméra que nous expérimentons sont le modèle de distorsion polynomiale classique [Lavest+98], [Sturm+11], [Lébraly+11] et le modèle de caméra unifié [Geyer+00].

Deuxièmement, nous appliquons le SfM monoculaire [Mouragnon+07], [Mouragnon+09] et un raffinement d'étalonnage par AF pour chaque caméra. Cependant, le SfM peut échouer sur une vidéo à cause de la combinaison de deux difficultés : manque de texture et étalonnage approximatif. Nous supposons qu'il existe au moins une vidéo sur laquelle les SfM et AF réussissent. Étant donné que les caméras ont le même réglage, nous bénéficions des paramètres intrinsèques raffinés par l'AF pour refaire le SfM monoculaire des autres vidéos. Ainsi, une difficulté (étalonnage approximatif) est réduite pour les vidéos les moins texturées et le risque d'échec diminue.

Troisièmement, une synchronisation à précision entière entre toutes les vidéos est obtenue en utilisant la méthode dans la Section I.4. A partir de là, on saute quelques images au début de chaque vidéo, de sorte que les restes des vidéos sont synchronisées à précision "trame" : dorénavant dans ce résumé, les images avec le même index sont prises en même temps modulo l'inverse du nombre d'images par seconde (ou frame per second en anglais - FpS).

Quatrièmement, une calibration de multi-caméra centrale est initialisée à partir des paramètres intrinsèques monoculaires estimés et des rotations inter-caméra approximatives (Section I.5). Nous appliquons le SfM multi-caméra [Mouragnon+07], [Mouragnon+09] suivi de l'AF multi-caméra [Lébraly+11] en ajoutant les paramètres intrinsèques comme nouveaux paramètres estimés. Jusqu'à maintenant, nous avons fait trois approximations : global shutter, multi-caméras centrale et décalages temporels sous-trame à zéro. En outre, nous avons seulement appliqués les SfM (à la fois monoculaire et multi-caméras) au début des vidéos pour obtenir la synchronisation et l'étalonnage initiaux (les 2k premières trames dans nos expériences). Ensuite, le SfM multi-caméras est appliqué une deuxième fois sur les vidéos complètes (sauf les quelques images au début sautées).

Enfin, nous appliquons l'AF multi-caméra dans la Section I.6 pour estimer la synchronisation à précision sous-trame et le retard de ligne (line delay) avec les paramètres habituels dont les paramètres intrinsèques.

I.3 Initialisation en caméra équiangulaire

Dans notre travail, nous utilisons deux modèles de caméra monoculaires. Les deux impliquent une matrice de paramètres intrinsèques `K` d'une caméra perspective : les distances focales f_x, f_y, et les coordonnées en pixel du point principal $\mathbf{p}_0(u_0, v_0)^\top$ dans l'image.

Les déformations géométriques engendrées par l'objectif d'une caméra peuvent être modélisées par des distorsions radiales (et tangentielles) - le modèle de distorsion polynomial classique [Lavest+98], [Sturm+11], [Lébraly+11]. Ce modèle est souvent utilisé pour les calculs de SfM et de géométrie épipolaire (pour apparier) car sa projection inverse est explicite. Le point image non-distordu (et le rayon) peut s'obtenir par une transformation polynomiale du point image distordu (original) $\mathbf{p}$. En pratique, on choisit un polynôme de degré l à coefficients k_i en r_d^2, où r_d est la distance radiale entre l'axe principal et le point distordu et les k_i sont les paramètres de distorsion radiale ($0 \leq i \leq l$). Ce modèle peut être appliqué à des caméras grand public telle que Gopro [GoPro]. Mais, celui-ci ne permet pas d'obtenir directement le point image distordu $\mathbf{p}$ car il n'y a pas de formule générale permettant d'exprimer les racines pour un polynôme de degré $l = 5$ (ou plus). Une méthode utilisée est celle de Gauss-Newton (ou Newton) qui a la particularité de converger très vite vers la solution de l'équation.

Le deuxième modèle est le modèle unifié de caméra [Geyer+00] qui est aussi intéressant car il modélise les caméras grand angle (fisheye) ayant un champ de vue supérieur à 180^o (comme celles des caméras sphériques [ThetaS]) bien qu'il ait un seul paramètre de distorsion radiale ξ. Pour cela, la projection de ce modèle est la composée de deux projections : la projection sur la sphère unité (par rapport à son centre), puis la projection par une caméra perspective de paramètres intrinsèques `K`. La projection sur la

sphère unité permet de modéliser les non-linéarités induites par l'objectif d'une caméra. Contrairement au modèle précédant modélisant la distorsion par un polynôme, ce modèle est "inversible" au sens où il est possible de retrouver analytiquement le point sur la sphère unité à partir du point image $\mathbf{p}$ et vice versa.

On initialise en caméra équiangulaire ces deux modèles de caméra. Une caméra est équiangulaire si l'angle μ entre l'axe principal et le rayon de projection inverse est proportionnel à la distance radiale non-normalisée $r_d = ||\mathbf{p} - \mathbf{p}_0||$ dans l'image originale. Dans ce cas, les pixels sont carrés (i.e. $f_x = f_y = f$) et des paramètres de distorsions approximatifs sont obtenus en utilisant un développement de Taylor. En pratique, nous initialisons $\mathbf{p}_0$ au centre de l'image et prenons $f = f(r_d(\mathbf{p}_1), \mu)$ pour un pixel $\mathbf{p}_1$ au centre d'un bord d'image où le demi-champ de vue μ est connu approximativement. D'abord, on effectue le SfM [Mouragnon+07], [Mouragnon+09] sur une des vidéos monoculaire avec cette calibration équiangulaire. Ensuite, cette calibration est raffinée par AF global. L'AF global raffine tous les paramètres : les paramètres intrinsèques, les poses de caméra et la géométrie (les points 3D). Pour les autres vidéos, on refait le SfM avec la calibration raffinée (résumé dans la Section I.2).

I.4 Synchronisation

L'initialisation de la synchronisation a deux étapes. Tout d'abord, on calcule les vitesses angulaires instantanées (IAV ou instantaneous angular velocity en anglais) par SfM monoculaire et un AF avec l'approximation global shutter. Toutes les trames sont reconstruites en utilisant le SfM basé sur des images clefs suivi par un calcul de pose pour toutes les images non clefs. Enfin, les vidéos sont synchronisées par un alignement de leurs vitesses angulaires. En pratique, on ne reconstruit que les 2000 premières (cela suffit pour synchroniser).

Soit $\mathtt{R}_{w_i,i}^{t_i}$ la matrice de passage du repère monde (w_i) au repère de la i-ème caméra pour sa t_i-ème image, i.e. la rotation de la caméra i à l'image t_i calculée par SfM monoculaire. L'IAV $\theta_i^{t_i}$ à l'instant t_i (pour la t_i-ème image) de la i-ème vidéo est approchée par l'angle de $\mathtt{R}_{w_i,i}^{t_i+1}(\mathtt{R}_{w_i,i}^{t_i})^\top$, qui est

$$\theta_i^{t_i} = \arccos\left(\frac{\mathrm{trace}(\mathtt{R}_{w_i,i}^{t_i+1}(\mathtt{R}_{w_i,i}^{t_i})^\top) - 1}{2}\right). \tag{I.1}$$

Soit $\mathtt{R}_{w_j,w_i}$ la matrice de passage du repère monde w_j au repère monde w_i et $\mathtt{R}_{j,i}$ la pose relative entre deux caméras i et j. Les deux sont constantes : la première est évidente et la dernière est aussi car les caméras sont rigidement liées les unes aux autres. Supposons que l'image t_i de la caméra i et l'image t_j de la caméra j sont prises au même instant. En calculant $\mathtt{R}_{w_i,i}^{t_i}$ de deux façons différentes, on obtient

$$\mathtt{R}_{w_j,w_i}\mathtt{R}_{w_i,i}^{t_i} = \mathtt{R}_{w_j,j}^{t_j}\mathtt{R}_{j,i}. \tag{I.2}$$

On a aussi

$$\mathtt{R}_{w_j,w_i}\mathtt{R}_{w,i}^{t_i+1} = \mathtt{R}_{w_j,j}^{t_j+1}\mathtt{R}_{j,i} \tag{I.3}$$

car les fréquences des vidéos sont identiques. Donc

$$\mathtt{R}_{w_j,j}^{t_j+1}(\mathtt{R}_{w_j,j}^{t_j})^\top = \mathtt{R}_{w_j,w_i}\mathtt{R}_{w_i,i}^{t_i+1}(\mathtt{R}_{w_i,i}^{t_i})^\top\mathtt{R}_{w_j,w_i}^\top. \tag{I.4}$$

Comme trace($\mathtt{XY}$) = trace($\mathtt{YX}$), on obtient

$$\text{trace}\left(\mathtt{R}_{w_j,j}^{t_j+1}(\mathtt{R}_{w_j,j}^{t_j})^\top\right) = \text{trace}\left(\mathtt{R}_{w_i,i}^{t_i+1}(\mathtt{R}_{w_i,i}^{t_i})^\top\right) \tag{I.5}$$

Nous voyons maintenant que $\theta_j^{t_j} = \theta_i^{t_i}$, i.e. toutes les caméras ont ce même angle au même instant (c'est la vitesse angulaire instantanée de la multi-caméra). On synchronise alors 2 caméras de la manière suivante : on calcule la table des θ_i^t pour chaque caméra i, puis on recherche le décalage entier o (offset) entre tables qui maximise un score de corrélation (ZNCC) des tables θ_i et θ_j

$$\text{ZNCC}(o_{i,j}) = \frac{\sum_t \left((\theta_i^t - m_i)(\theta_j^{t+o_{i,j}} - m_j)\right)}{\sqrt{\sum_t (\theta_i^t - m_i)^2}\sqrt{\sum_t (\theta_j^{t+o_{i,j}} - m_j)^2}} \tag{I.6}$$

où m_i et m_j sont les moyennes de tables θ_i et θ_j.

On peut raffiner o à précision "sous-trame" de la façon suivante. La fonction f qui à $\epsilon \in [-1, +1]$ associe le score de corrélation pour le décalage $o + \epsilon$ est connue en $\epsilon \in \{-1, 0, 1\}$. On approxime f par un polynôme de degrés 2 connaissant ses valeurs en $\{-1, 0, 1\}$, et ϵ maximise ce polynôme.

En pratique, on a plus de 2 caméras et on estime les décalages entiers entre caméras adjacentes i et $i + 1$, ex : $o_{0,1}, o_{1,2}, o_{2,3}$, $o_{3,0}$ dans le cas de 4 caméras. Soit $L = o_{0,1} + o_{1,2} + o_{2,3} + o_{3,0}$. L'objectif de la synchronisation est de passer les s_i premières images de la i-ème vidéo, de sorte que les vidéos soient synchronisées pour le SfM et l'AF multi-caméra. On a donc 4 relations $s_i - s_{i+1} = o_{i,i+1}$ (indices modulo 4), et ceci n'est possible que si $L = 0$. Comme les décalages sont estimés de façon indépendante, on peut avoir $L \neq 0$. Dans ce cas, on cherche à remplacer chaque décalage o par un autre dans $\{o-k, \ldots, o-1, \ldots, o+k\}$ de sorte à maximiser la somme des corrélations (pour chaque o) sous la contrainte $L = 0$ ($k = 1$ suffit en pratique). Cette méthode se généralise à n caméras (considérer tous les $o_{i,j}, i \neq j$ et toutes les boucles d'adjacentes $i \to j \to l \to i$).

I.5 Auto-étalonnage avec l'hypothèse de global shutter et synchronisation à précision trame

Soient $n \geq 2$ caméras rigidement liées, dont les paramètres intrinsèques $\mathbf{I}^j (j \in \{0, 1, \ldots, n-1\})$ sont initialisés par la méthode dans la Section I.3. Les caméras sont synchronisée à précision trame et se déplacent le long d'un parcours en observant un nuage de points 3D $\mathcal{X}$. Soient $m + 1$ le nombre de poses du système multi-caméra. Chaque pose du système multi-caméra est représentée par la transformation homogène $\mathtt{T}_{\text{M}}^i$ $(i \in \{0, 1, \ldots, m\})$ dans le repère monde. De même, $\mathtt{T}_{\text{C}}^j$ est la transformation homogène passant du repère de multi-caméra à celui de la j-ème caméra (elle est également appelée paramètres extrinsèques). Chaque transformation homogène $\mathtt{T}$ est exprimée à l'aide d'une rotation $\mathtt{R}$ et d'une translation $\mathbf{t}$. La Figure I.2 illustre géométriquement l'ensemble du système, ainsi que les paramètres optimisés.

La méthode suivante permet d'obtenir conjointement l'étalonnage (extrinsèque et intrinsèque) du système multi-caméra, ses poses et la reconstruction d'une carte 3D d'amers visuels dans un repère monde.

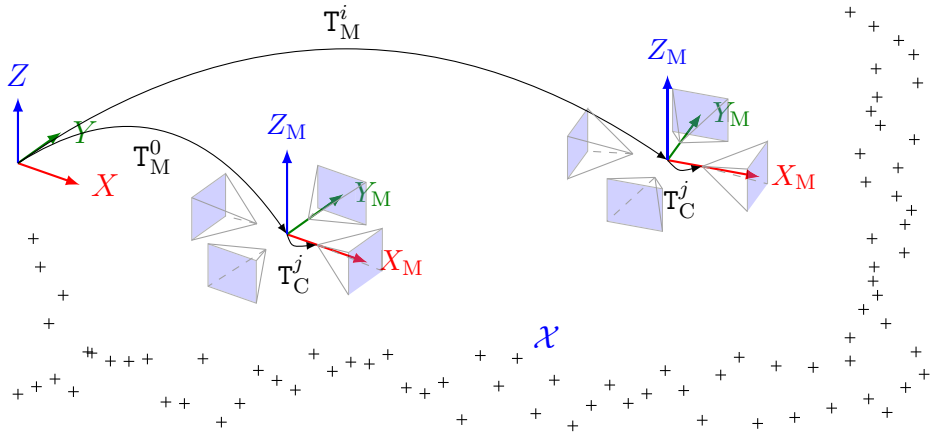


FIGURE I.2: Système multi-caméra se déplaçant le long d'une scène statique.

- A partir d'une connaissance approximative sur les paramètres extrinsèques $\mathtt{T}_{\mathrm{C}}^{j}$ (ou d'une estimation initiale par des méthodes antérieures), un algorithme de SfM multi-caméra [Mouragnon+07], [Mouragnon+09] est appliqué. On obtient alors une reconstruction grossière.
- Les paramètres extrinsèques et intrinsèques, les poses de la multi-caméra et les points 3D sont optimisés par l'AF multi-caméra (MCBA pour multi-camera bundle adjustment).

La Figure I.3 illustre une vue d'ensemble de l'algorithme d'étalonnage proposé.

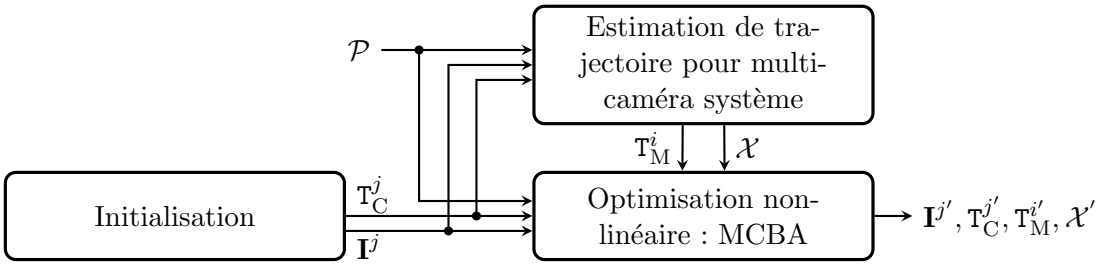


FIGURE I.3: Vue d'ensemble de l'étalonnage de système multi-caméra. $\mathcal{P}$ est l'ensemble de points 2D détectés et appariés dans des images clefs sélectionnées. $\mathcal{X}$ est l'ensemble de points 3D reconstruits.

On décrit maintenant le MCBA. Soit $\tilde{\mathbf{p}}$ l'observation d'un point 3D $\mathbf{X}^{l} \in \mathcal{X}$ détectée dans la sous-image numéro j (prise par la j-ème caméra) de l'image-clef i et $\mathbf{p}$ le point projeté de $\mathbf{X}^{l}$ en changeant de repère de coordonnées 3D deux fois (d'abord en passant du repère monde au repère multi-caméra avec $\mathtt{T}_{\mathrm{M}}^{i}(\mathtt{R}_{\mathrm{M}}^{i}, \mathbf{t}_{\mathrm{M}}^{i})$ puis en passant du repère multi-caméra au repère caméra avec $\mathtt{T}_{\mathrm{C}}^{j}(\mathtt{R}_{\mathrm{C}}^{j}, \mathbf{t}_{\mathrm{C}}^{j})$), puis en appliquant la fonction de projection (dépendant des paramètres intrinsèques $\mathbf{I}^{j}$). Donc, la projection $\mathbf{p}$ de $\mathbf{X}^{l}$ est une fonction

$$\mathbf{p} = f(\mathbf{I}^{j}, \mathtt{R}_{\mathrm{C}}^{j}, \mathbf{t}_{\mathrm{C}}^{j}, \mathtt{R}_{\mathrm{M}}^{i}, \mathbf{t}_{\mathrm{M}}^{i}, \mathbf{X}^{l}). \tag{I.7}$$

Soit $\boldsymbol{\epsilon}_{ij}^{l} = \mathbf{p} \quad \tilde{\mathbf{p}}$ l'erreur de reprojection du point $\mathbf{X}^{l}$ observé par la caméra j à la i-ème pose dans l'espace image original (distordu). La fonction de coût que l'on cherche à

minimiser est

$$F = \frac{1}{2} \sum_{(i,j,l)} (\boldsymbol{\epsilon}_{ij}^{l})^{\top} \boldsymbol{\epsilon}_{ij}^{l} = \frac{1}{2} \boldsymbol{\epsilon}^{\top} \boldsymbol{\epsilon} \tag{I.8}$$

où $\boldsymbol{\epsilon}$ est le vecteur résidu qui concatène les erreurs de reprojection $\boldsymbol{\epsilon}_{ij}^{l}$. On utilise l'algorithme de Levenberg-Marquardt (LM) [Madsen+04], [Agarwal+] pour minimiser la fonction de coût F. Il s'agit d'une optimisation non-linéaire itérative qui résout un problème de moindres carrés en combinant les advantages des optimisations de type descente de gradient et de Gauss-Newton.

Les rotations sont paramétrées localement par des angles d'Euler incrémentaux classiques [Triggs+00]. Une implémentation efficace de l'AF multi-caméra utilise à la fois la structure creuse et l'expression analytique de la matrice Jacobienne des erreurs de reprojection en fonction des paramètres à optimiser. Notre méthode est une amélioration de l'algorithme proposé par [Lébraly+11] : on raffine en plus les paramètres intrinsèques, et on minimise les erreurs de reprojection dans l'espace image original (de mesure/détection) pour le modèle distorsion polynomial classique (Section I.3). Sous l'hypothèse d'un bruit gaussien des points 2D détectés de moyennes nulles, indépendants et des distributions normales, notre MCBA est un estimateur au maximum de vraisemblance. Cette hypothèse n'est pas vraie dans l'espace d'image rectifié. L'extension pour le modèle de caméra unifié (dans (Section I.3)) est simple.

I.6 Auto-étalonnage avec l'hypothèse de rolling shutter et synchronisation à précision sous-trame

Une caméra global shutter (GS) expose toutes les lignes simultanément, donc toutes les lignes ont la même pose. En revanche, pour une caméra rolling shutter (RS), les différentes lignes sont acquises de manière séquentielle. Lorsque la caméra se déplace, chaque ligne de pixels a une pose différente. De plus, dans certains cas, par exemple notre multi-caméra formée de quatre GoPro, les caméras monoculaires enregistrent les vidéos séparément. Même si toutes les caméras sont synchronisées à précision trame (FA ou frame-accurate en anglais), il y a encore les décalages sous-trame entre caméras. La synchronisation à précision sous-trame et le rolling shutter (Figure I.4) compliquent notre calibration pour la même raison : une calibration variable non-centrale. On propose une méthode spécifique qui estime non seulement les paramètres habituels (les paramètres intrinsèques et extrinsèques, les poses de caméra, les points 3D) mais aussi la synchronisation et le coefficient de RS. Ici le formalisme diffère de la Section I.5 où l'on a fait l'hypothèse de synchronisation FA et approximation GS.

Soient $n \geq 2$ caméras rigidement liées, initialement synchronisées (à précision trame) se déplaçant le long d'une trajectoire. Les paramètres intrinsèques et extrinsèques $(\mathtt{R}_{\mathrm{C}}^{j}, \mathbf{t}_{\mathrm{C}}^{j})$ (avec $0 \leq j \leq n-1$) sont initialisés en utilisant la méthode dans la Section I.5 sur le début des vidéos (les 2k premières images dans nos expériences). Le SfM avec l'hypothèse GS [Mouragnon+09] est appliqué sur les séquences complètes, génère $m+1$ images clés et estime leurs poses $\mathbf{m}_i(\mathtt{R}_{\mathrm{M}}^{i}, \mathbf{t}_{\mathrm{M}}^{i})$ (avec $0 \leq i \leq m$). La pose d'une caméra peut changer durant l'acquisition d'une image complète : l'exposition des lignes (de pixels) différentes de l'image n'est pas synchrone. Lors de la projection d'un point sur un pixel, il est nécessaire de connaître la trajectoire de caméra au moment de l'exposition de ce pixel. La notation de pose doit donc être étendue pour prendre en compte cette

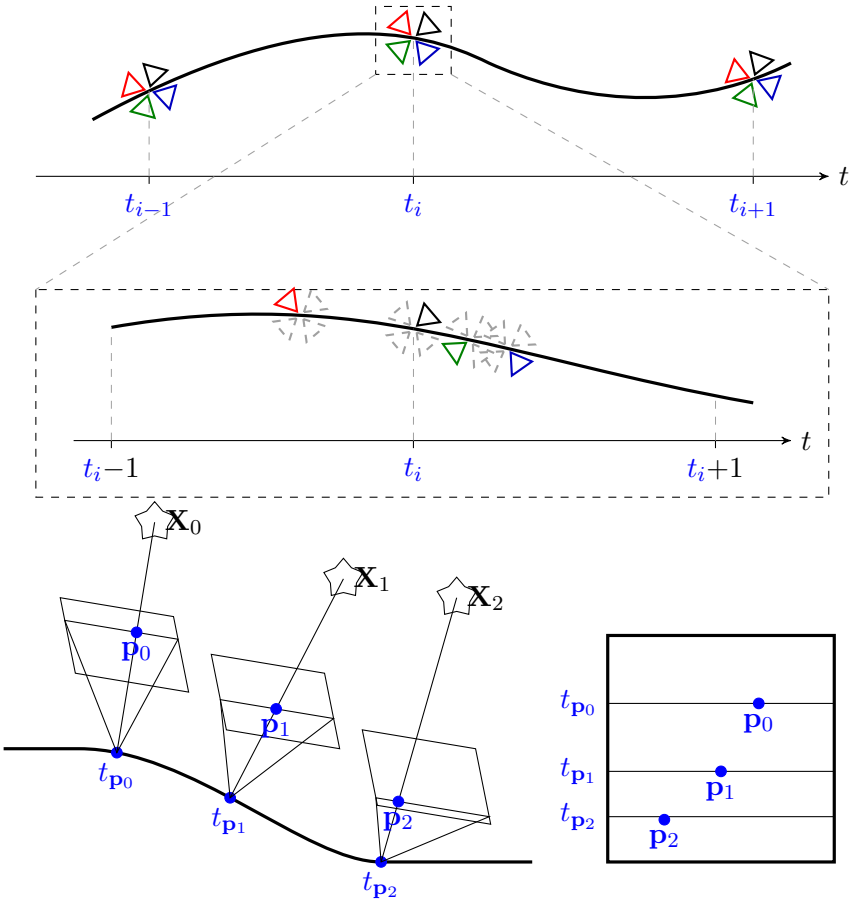


FIGURE I.4: Trajectoire continue en temps d'une multi-caméra. En haut, quatre caméras monoculaires à l'instant t_i, qui ont des décalages temporels non nulles. En bas, une caméra RS monoculaire, qui se déplace et observe des points à l'instant différent dan une seule image. Différentes lignes d'une image ont différentes poses.

dépendance au temps. Cela nécessite de nouveaux paramètres, incluant ceux des poses clés $\mathbf{m}_i$, qui définissent les variations de trajectoire de caméra durant l'acquisition.

La Section I.6.1 présente des nouveaux paramètres qui prennent en compte la synchronisation à précision sous-trame et le rolling shutter. Un modèle de trajectoire de multi-caméra est introduit dans la Section I.6.2. La Section I.6.3 décrit la projection d'image.

I.6.1 Paramétrage

Dans le cas d'une image global shutter, il n'a qu'une pose puisque l'ensemble des pixels de l'image sont exposés au même instant. Pour une image rolling shutter, afin de pouvoir projeter un point 3D sur un pixel de l'image avec la trajectoire de caméra, il est nécessaire que cette pose $M(t)$ soit définie à l'instant t où le pixel est exposé. Soit $\mathcal{R}$ une fonction de classe $\mathcal{C}^1$ surjective qui à tout vecteur dans $\mathbb{R}^k$ associe une rotation dans l'ensemble $\mathbb{SO}(3)$. Pour donner un sens plus précis, il faut s'intéresser à la paramétrisation de pose $M(t)^\top = \left(\mathbf{t}_\mathrm{M}(t)^\top \mathbf{r}_\mathrm{M}(t)^\top\right)$ où $\mathbf{t}_\mathrm{M}(t) \in \mathbb{R}^3$ est la translation et $\mathcal{R}(\mathbf{r}_\mathrm{M}(t)) \in \mathbb{SO}(3)$ est la rotation. Pour la rotation, on utilise des angles d'Euler

(globaux) sous forme de vecteurs à trois coordonnées ou des quaternions sous forme de vecteurs à quatre coordonnées (donc $k = 3$ ou $k = 4$). Pour des angles d'Euler, on évite les cas de singularité en considérant une classe de trajectoires de multi-caméra : tous les mouvements de lacet sont possibles, mais ceux de tangage et de roulis sont petits. De tels mouvements sont très fréquents pour une multi-caméra fixée sur un casque et un utilisateur explorant l'environnement sans objectif spécial, comme la saisie l'objet sur le terrain (et aussi pour une multi-caméra fixée sur un véhicule). Pour éviter ces singularités, on change les repères monde et multi-caméra tels que l'angle de rotation selon l'axe *oy* est proche de 0 pour toutes les rotations d'images clés de la trajectoire de multi-caméra avant notre AF.

Comme la caméra est RS, le temps τ séparant l'exposition de lignes adjacentes est tel que la v-ème ligne de la première sous-image dans la i-ème trame clé est prise au temps $t_i + v\tau$. De plus, nous avons une multi-caméra. La i-ème trame clé de cette multi-caméra est composée de sous-images prises par les caméras monoculaires (Figure I.5). Soit $\Delta_j \in \mathbb{R}$ le décalage réel entre la j-ème vidéo et la première vidéo. La première ligne de la j-ème sous-image dans la i-ème trame clé est donc prise à l'instant $t_i + \Delta_j$. Comme toutes les caméras ont la même fréquence et le même décalage τ (constant), la v-ème ligne de la j-ème sous-image dans la i-ème trame clé est prise au moment

$$t(v) = t_i + \Delta_j + v\tau. \tag{I.9}$$

On prend $\mathbf{m}_i = M(t_i)$, ce qui correspond à la pose lors de l'exposition t_i de la première ligne de la première sous-image de la trame clé i. En utilisant les poses clés $\mathbf{m}_i$ voisines, des approches pour estimer une pose $M(t)$ à l'instant t sont présentées dans la prochaine partie.

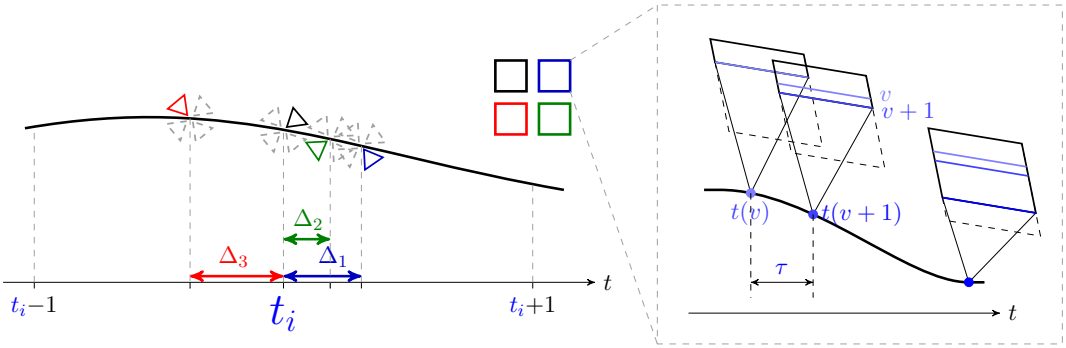


FIGURE I.5: Paramètres de temps, rolling shutter et synchronisation. A gauche, Δ_j est le décalage réel entre la j-ème vidéo et la première vidéo. A droite, τ est le décalage entre l'exposition de deux lignes successives. La caméra se déplace pendant l'exposition d'une image.

I.6.2 Trajectoire de multi-caméra

On propose une approximation dans l'espace Euclidien $\mathbb{R}^k$ qui utilise des développement de Taylor, et dont on peut calculer les dérivées analytiques facilement. On considère aussi deux interpolations sphériques (*Squad* pour les quaternions unitaires [Dam+98] et spline [Jakubiak+06]). Les deux dernières sont plus complexes, y compris pour les dérivées. Dans la cadre de ce résumé, on présente seulement l'approximation linéaire

$M_1(t)$ de $M(t)$ dans l'espace Euclidien qui non seulement est efficace au niveau d'implémentation, mais aussi donne des résultats au moins aussi précis que des interpolations sphériques dans la majorité de nos expériences. Plus de détails sur ces interpolations sphériques sont présentés dans les chapitres précédents de ce mémoire.

On a le développement de Taylor :

$$M(t) = \mathbf{m}_i + (t - t_i)\dot{M}(t_i) + \mathcal{O}(|t - t_i|^2). \tag{I.10}$$

Soit $\delta = \max_i(t_{i+1} - t_i)$. La dérivée de M par rapport à t en t_i est une moyenne pondérée de 3 poses contrôles voisines $\mathbf{m}_{i-1}, \mathbf{m}_i, \mathbf{m}_{i+1}$ (voir Figure I.6) :

$$\dot{M}(t_i) = D_1(\mathbf{m}_{i-1}, \mathbf{m}_i, \mathbf{m}_{i+1}, t_{i+1} - t_i, t_i - t_{i-1}) + \mathcal{O}(\delta^2) \tag{I.11}$$

où

$$D_1(\mathbf{x}, \mathbf{y}, \mathbf{z}, a, b) = -\frac{a\mathbf{x}}{b(a+b)} + \frac{(a-b)\mathbf{y}}{ab} + \frac{b\mathbf{z}}{a(a+b)}. \tag{I.12}$$

La valeur de M à l'instant t est approchée par le développement de Taylor en t_i en négligeant les grands $\mathcal{O}$, pour $0 < i < m$,

$$M_1(t) = \mathbf{m}_i + (t - t_i)D_1(\mathbf{m}_{i-1}, \mathbf{m}_i, \mathbf{m}_{i+1}, t_{i+1} - t_i, t_i - t_{i-1}) \tag{I.13}$$

si $t \approx t_i$. Pour $i = 0$ (et similairement pour $i = m$), on utilise $M(t) = \mathbf{m}_0 + (t - t_0)(\mathbf{m}_1 - \mathbf{m}_0)/(t_1 - t_0)$. De la même manière, on obtient une approximation quadratique M_2 de M.

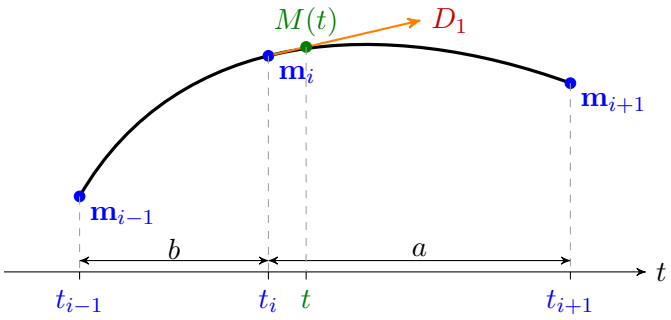


FIGURE I.6: Approximation linéaire M_1 de M.

I.6.3 Projection dans une image

Étant donné que notre MCBA minimise la somme du module au carré d'erreur de reprojection pour chaque inlier, cette section décrit le calcul d'une erreur de reprojection d'un point 3D (dans le repère monde) et son observation $\tilde{\mathbf{p}}$ dans la sous-image numéro j de l'image-clef i. Tout d'abord, on introduit des notations. Soit $\mathbf{p}$ la projection du point 3D $\mathbf{X}$ dans la sous-image numéro j de l'image-clef i. L'erreur de reprojection est $\mathbf{p} - \tilde{\mathbf{p}}$. On rappelle que $(\mathtt{R}_{\mathrm{C}}^j, \mathbf{t}_{\mathrm{C}}^j)$ est la pose de caméra j dans le repère multi-caméra. Soit p_j la fonction de projection de la caméra j. On suppose que les paramètres intrinsèques impliqués dans p_j et $\mathtt{R}_{\mathrm{C}}^j, \mathbf{t}_{\mathrm{C}}^j$ sont constant. Les points $\mathbf{p}(u, v)^\top$ et $\tilde{\mathbf{p}}(\tilde{u}, \tilde{v})^\top$ sont acquis aux instants :

$$t_{\mathbf{p}} = t_i + \Delta_j + v\tau \quad \text{et} \quad t_{\tilde{\mathbf{p}}} = t_i + \Delta_j + \tilde{v}\tau. \tag{I.14}$$

Deuxièmement, on détaille la relation entre $\mathbf{p}$ et $\mathbf{X}$. La pose $(\mathbf{r}_{\mathrm{M}}(t_{\mathbf{p}}), \mathbf{t}_{\mathrm{M}}(t_{\mathbf{p}}))$, i.e. $M(t_{\mathbf{p}})$, est définie dans la Section I.6.2 en utilisant l'indice i d'image clé et $t = t_{\mathbf{p}}$. Les coordonnées de $\mathbf{X}$ dans le repère multi-caméra sont dans le vecteur

$$\mathbf{X}_{\mathrm{M}} = \begin{bmatrix} \mathcal{R}^{\top}(\mathbf{r}_{\mathrm{M}}(t_{\mathbf{p}})) & -\mathcal{R}^{\top}(\mathbf{r}_{\mathrm{M}}(t_{\mathbf{p}}))\mathbf{t}_{\mathrm{M}}(t_{\mathbf{p}}) \\ \mathbf{0}^{\top} & 1 \end{bmatrix} \mathbf{X}. \tag{I.15}$$

Les coordonnées de $\mathbf{X}$ dans le repère de la caméra j et la projection de $\mathbf{X}$ sont

$$\mathbf{X}_{\mathrm{C}} = \begin{bmatrix} (\mathtt{R}_{\mathrm{C}}^{j})^{\top} & -(\mathtt{R}_{\mathrm{C}}^{j})^{\top}\mathbf{t}_{\mathrm{C}}^{j} \\ \mathbf{0}^{\top} & 1 \end{bmatrix} \mathbf{X}_{\mathrm{M}} \quad \text{et} \quad \mathbf{p} = p_j(\mathbf{X}_{\mathrm{C}}). \tag{I.16}$$

On voit que le calcul de $\mathbf{p}$ a besoin du calcul de $\mathbf{X}_{\mathrm{M}}$ qui à son tour nécessite le calcul du point 2D $\mathbf{p}$.

Ce problème peut-être résolu grâce à une approximation [Klingner+13] : $t_{\mathbf{p}}$ est remplacé par $t_{\tilde{\mathbf{p}}}$ dans Eq.(I.15). Ceci est acceptable car l'observation $\tilde{\mathbf{p}}$ est inlier et l'ordre de grandeur de τ est très petit : $|t_{\tilde{\mathbf{p}}} - t_{\mathbf{p}}| \leq \tau||\mathbf{p} - \tilde{\mathbf{p}}||$ et $\tau \approx 10^{-5}$ (s/pixel) et $||\mathbf{p} - \tilde{\mathbf{p}}|| \leq 4$ pixels. On propose aussi le calcul exact de $\mathbf{p}$ sans cette approximation, c'est-à-dire sans $t_{\tilde{\mathbf{p}}} = t_{\mathbf{p}}$. Soit $\boldsymbol{\theta}$ le vecteur de paramètres optimisés par AF. On sait que $\tilde{\mathbf{p}}$ est proche de $\mathbf{p}$ car on ne considère que les inliers. On connaît aussi une fonction de classe $\mathcal{C}^1$ $g(\mathbf{p}, \boldsymbol{\theta})$ de $\mathbb{R}^2 \times \mathbb{R}^p$ à $\mathbb{R}^2$ tel que $\mathbf{p}$ est la solution $\mathbf{z}$ de $g(\mathbf{z}, \boldsymbol{\theta}) = \mathbf{0}$ (grâce aux Eqs.(I.14,I.15,I.16)), et la valeur courante $\boldsymbol{\theta}_0$ de $\boldsymbol{\theta}$ (fournie par l'initialisation ou l'itération précédente d'AF). D'abord, $\mathbf{p}$ est estimé par une méthode de moindres carrés non-linéaire en minimisant $\mathbf{z} \mapsto ||g(\mathbf{z}, \boldsymbol{\theta}_0)||^2$. En pratique, on utilise la méthode itérative de Gauss-Newton à partir de $\mathbf{z} = \tilde{\mathbf{p}}$ avec pas plus de 5 itérations. Les dérivées de $\mathbf{p}$ par rapport à tous les paramètres sont obtenues par les théorèmes des fonctions implicites et de dérivation des fonctions composées.

I.7 Expérimentations

Pour évaluer les performances de nos méthodes, on a effectué un certain nombre d'expérimentations sur des jeux de données : synthétique et réelles avec différents types de multi-caméras. Elles ont un champ de vue de 360^o dans le plan horizontal et se déplacent dans des environnements urbains ou ruraux.

I.7.1 Notations

Notre MCBA est caractérisée par une combinaison de plusieurs notations qui décrivent les paramètres estimés :

- C (l'approximation centrale) estime toutes les rotations $\mathtt{R}_{\mathrm{C}}^{j}$ et fixe toutes les translations $\mathbf{t}_{\mathrm{C}}^{j} = \mathbf{0}$,
- NC (non-centrale) estime toutes les rotations et translations $(\mathtt{R}_{\mathrm{C}}^{j}, \mathbf{t}_{\mathrm{C}}^{j})$,
- INT (intrinsèque) estime tous les paramètres intrinsèques : $(f_x, f_y, u_0, v_0, k_1, \ldots, k_5)$ ou $(f_x, f_y, u_0, v_0, \xi)$ selon le modèle de caméra choisi ; chaque caméra a ses propres paramètres,

- FA (frame accurate) fixe tous les décalages $\Delta_j = 0$,
- SFA (sub-frame accurate) estime tous les Δ_j,
- GS (global shutter) fixe le line delay $\tau = 0$,
- RS (rolling shutter) estime τ.

Donc, GS.NC.SFA.INT (ou gs.nc.sfa.int) est un AF qui fixe $\tau = 0$ et estime simultanément tous les paramètres $\Delta_j, \mathtt{R}_\mathrm{C}^j, \mathbf{t}_\mathrm{C}^j$, les paramètres intrinsèques, les poses clés $\mathbf{m}_i$ et les points 3D. Le seuil d'inlier est de 4 pixels pour tous les cas. Chaque AF a trois mise à jour d'inliers chacune étant suivie par la minimisation de Levenberg-Marquardt.

I.7.2 Systèmes multi-caméras utilisés

Nous utilisons quatre multi-caméras (voir Figure I.1). Les deux premières sont composées de quatre caméras Gopro Hero 3 [GoPro] qui sont rolling shutter, fixées sur un casque. Il y a deux configurations différentes : les caméras sont enfermées dans une boîte en carton (avec une baseline faible) et prennent des images 1280×960 à 100Hz ; ou bien elles sont fixées dans leur coques fournies par le constructeur (la baseline entre les caméras est plus grande) et prennent des images 1920×1440 à 48Hz. La troisième est une caméra à 360^o bas de gamme - la Ricoh Theta S [ThetaS] qui est modélisable par deux caméras fisheyes synchronisées avec un champ de vue plus large que 180^o pour chacune. La dernière (Ladybug 2 [Ladybug2]) est parfaitement calibrée et synchronisée. Elle est utilisée pour l'évaluation quantitative. Elle est composée de cinq caméras global shutter prenant des images 1024×768 à 15Hz (il y en a une 6-ème pointant vers le haut, mais on ne l'utilise pas dans nos expériences). A l'exception des cas synthétique et Ladybug, les multi-caméras ont des vérités terrains incomplètes (un stroboscope fournit toujours la valeur de τ). Les quatre caméras Gopro et Theta S sont montées sur un casque. La caméra Ladybug est fixée sur une voiture grâce à un mât.

I.7.3 Jeux de données

Le jeu de données synthétique est un ensemble d'images généré à partir d'un modèle 3D existant de milieu urbain (la vérité terrain de la calibration est complètement connue, y compris la synchronisation et le coefficient de RS). Les jeux de données réels sont pris par nos multi-caméras qui se déplacent le long d'un parcours dans des environnements urbains et naturels. Ils sont acquis dans plusieurs contextes : caméra Ladybug fixée sur une plate-forme de voiture (CarCity - CC) et placée à environ 4m du sol, quatre caméras GoPro fixées sur un casque et se déplaçant à vélo (BikeCity1 - BC1) ou à pied (WalkTown - WT) dans des environnements urbains ou faisant du parapente au-dessus d'une colline (FlyHill - FH), caméra Ricoh Theta S se déplaçant à pied dans le campus de l'Université Clermont Auvergne (WalkUniv - WU). La Figure I.1 illustre tous les jeux de données. Le jeu de données réel le plus long est BC1 (2.5km) et le jeu de donnée synthétique est BC2. La longueur de la trajectoire est d'au moins 600m.

I.7.4 Comment évaluer la calibration ?

On aimerait, avec un unique nombre, évaluer l'erreur de notre calibration multi-caméra avec celle de la vérité terrain. On utilise une distance d basée sur les angles entre les directions de rayons de calibrations (estimée et vérité terrain) correspondant à un même pixel. Il y a plusieurs raisons pour faire cela. D'abord, on a seulement besoin de la direction des rayons pour le calcul de SfM central [Mouragnon+09]. Ensuite, on utilise la vérité terrain fournie par le constructeur de la multi-caméra (Ladybug), qui est une table de rayons. Enfin, des paramètres peuvent se compenser s'ils sont biaisés (ex : l'ambiguïté partielle rotation-point principal [Agapito+01] d'une caméra). Il faut aussi tenir compte du fait que les repères multi-caméras ne sont pas les mêmes. On estime pour cela la rotation $\mathtt{R}$ qui permet de passer d'un repère à l'autre. On minimise

$$e(\mathtt{R}) = \sum_{i=1}^{N} ||\mathbf{r}_i^{\text{vt}} - \mathtt{R}\mathbf{r}_i^{\text{est}}||^2 \tag{I.17}$$

avec $\mathbf{r}_i^{\text{vt}}$ et $\mathbf{r}_i^{\text{est}}$ qui sont les directions des rayons correspondants au même pixel pour les calibrations vérité terrain et estimée, respectivement. Puis, on définit

$$d = \sqrt{\frac{e(\mathtt{R})}{N}} \tag{I.18}$$

avec N le nombre de rayons (échantillonnés) dans l'image multi-caméra.

I.7.5 Synchronisation à précision trame

D'abord, on effectue le SfM [Mouragnon+07], [Mouragnon+09] sur une des vidéos monoculaires avec une calibration équiangulaire à champ de vue approximativement connu. Ensuite, cette calibration est raffinée par AF global. Pour les autres vidéos, on refait le SfM avec la calibration raffinée. Jusque là, seules des images clefs sont reconstruites, on effectue alors un calcul de pose pour toutes les images. En pratique, on ne calcule que les 2000 premières images de chaque vidéo. Enfin, on calcule les décalages entiers comme dans la Section I.4.

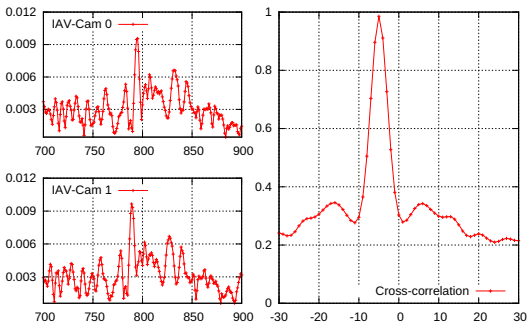
La Figure I.7 montre la vitesse angulaire instantanée et le score ZNCC pour les trois séquences BC1 (à vélo), WT (à pied) et CC (sur plate-forme de voiture). La Table I.1 montre les résultats pour tous les jeux de données avec la contrainte $L = 0$. Les décalages entiers sont bons : très proches de la vérité terrain pour CC, BC2 et WU. On remarque qu'il y a un décalage ± 15 non négligeable entre caméras (0.15 seconds) pour WT.

I.7.6 Calibration avec l'hypothèse de global shutter et synchronisation à précision trame

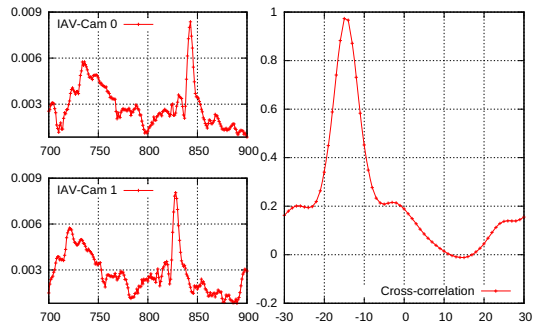
Une fois les décalages entiers calculés, on applique le SfM et l'AF multi-caméras pour estimer la calibration multi-caméra. On initialise la pose des caméras dans le repère multi-caméra : les rotations inter-caméras sont d'angles multiples de $2\pi/n$ autour de l'axe oz (pour $n = \{2, 4, 5\}$ caméras) et les centres sont en $(0, 0, 0)^\top$. Ici, on fait l'approximation centrale. Puis l'AF optimise les points 3D, les poses successives du système multi-caméra, et étalonne intrinsèquement et extrinsèquement les caméras

Nom	$o_{0,1}$	$o_{1,2}$	$o_{2,3}$	$o_{3,4}$	ZNCC_1	ZNCC_2
BC1	-5	3	4	na	3.912	3.884
WT	-15	-1	14	na	3.919	3.918
FH	-1	1	-2	na	3.991	3.983
BC2	0	0	0	na	3.915	3.907
CC	0	0	0	0	4.987	4.347
WU	0	na	na	na	0.993	0.677

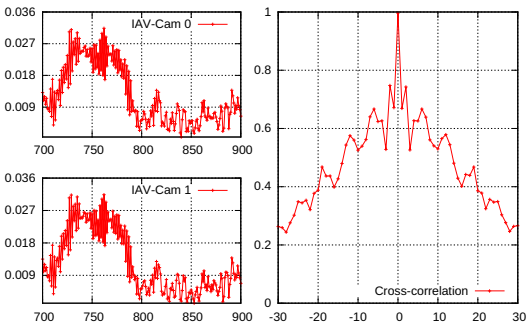
TABLE I.1: Décalages entiers $o_{j,j+1}$ avec contrainte de boucle. ZNCC_1 est le meilleur score de n décalages calculés, et ZNCC_2 est le deuxième meilleur score (donc $-n \leq \text{ZNCC} \leq +n$). On rappelle que $o_{j,j+1}$ compte un nombre signé de trames entres la vidéo j. Si la multi-caméra a n caméras numérotées de 0 à $n-1$, le décalage $o_{j,j+1}$ est "na" si $j+1 \geq n$.



(a) BC1



(b) WT



(c) CC

FIGURE I.7: Les vitesses angulaires instantanées et le score ZNCC pour cam0 et cam1 de BC1, WT et CC. A gauche, l'axe des abscisses est l'index de trame et l'axe des ordonnées est la vitesse angulaire instantanée en radian. A droite, on a les décalages candidats et leur corrélation ($\text{ZNCC} \in [-1, 1]$).

(avec l'hypothèse de global shutter et synchronisation à précision trame). On compare les calibrations obtenues entre minimisations dans les espaces images original et rectifié pour le modèle de distorsion polynomial classique (dans la Section I.3). La Table I.2 compare les résultats de notre AF en termes de RMS et d'erreur angulaire d. Si init $= 72r$, les paramètres intrinsèques initiaux sont ceux calculés par la synchronisation (on part d'une calibration équiangulaire que l'on raffine par AF global - d'où le "r"). Sinon, init $= pat$, les caméras sont calibrées avec une mire [Lavest+98] et l'AF ne raffine pas les paramètres intrinsèques. Dans tous les cas, on a des nombres d'inliers 2D similaires et l'erreur de calibration d dans l'espace distordu est meilleur (plus petite) que celle

dans l'espace rectifié. On explique cela de la façon suivante : il y a de grandes distorsions entre les images rectifiées et distordues, les AFs minimisent les erreurs dans les différents espaces (distordu et rectifié) ; et sous l'hypothèse habituelle que les bruits d'images sont de moyennes nulles, indépendants, et ont des distributions normales et identiques, notre AF est un estimateur au maximum de vraisemblance (cette hypothèse n'est pas vraie dans l'espace rectifié). On donne aussi le nombre d'inliers 2D et le RMS de GS.C.FA qui utilise les paramètres intrinsèques estimés avec une mire (ces paramètres sont fixés pendant l'AF) ; nos RMSs et inliers sont légèrement meilleur, mais l'erreur d de *pat* est la meilleure.

Erreur	init	Méthode	#2D	RMS	d
Rectifié	$72r$	gs.c.fa.int	213335	1.216	9.575
	pat	gs.c.fa	213015	1.225	1.023
Distordu	$72r$	gs.c.fa.int	213495	0.932	1.683
	pat	gs.c.fa	213108	0.946	1.023

TABLE I.2: Résultat d'AFs gs.fa.X pour les séquences courtes (2k) de CC : distance d (convertie en pixels en utilisant la résolution angulaire), nombre de points 2D inliers et RMS d'erreurs de reprojections en pixels.

I.7.7 Rolling shutter et synchronisation à précision sous-trame

Pour l'étape d'auto-étalonnage avec l'hypothèse de rolling shutter et synchronisation à précision sous-trame, les résultats sont également évalués quantitativement avec les vérités terrains. On calcule l'erreur $e(\Delta)$ qui est la somme de l'erreur absolue pour tous les $f\Delta_j$ (il y a $n-1$ valeurs), l'erreur relative $e(\tau)$ pour τ et l'erreur de calibration multi-caméra d.

I.7.7.1 Précision

La Table I.3 montre les résultats pour deux séquences BC2 et CC qui ont la vérité terrain pour τ et Δ_j. On compare les estimations séparée et simultanée de paramètres intrinsèques (INT) et rolling shutter et synchronisation sous-trame (RS.SFA), c.-à.-d. on compare GS.C.FA.INT+RS.C.SFA (le premier suivi du second) et RS.C.SFA.INT. On compare aussi la calibration centrale et non-centrale, et la synchronisation à précision sous-trame sans AF de la Section I.4. Pour le jeu de données synthétique BC2, la précision de τ et d dans l'estimation simultanée est meilleure que celle dans l'estimation séparée (dans tous les deux cas central ou non-central). Cependant, l'estimation séparée donne $e(\Delta)$ deux fois plus petit que celui de l'estimation simultanée qui à son tour est deux fois plus petit que celui de la synchronisation sans AF (Section I.4). L'AF non-central diminue l'erreur d. Pour le jeu de données réel CC, les caméras sont GS et parfaitement synchronisées. L'erreur relative $e(\tau)$ n'est donc pas un nombre et on le remplace par $v_{max}f\tau$ (la plus petite valeur absolue donne le meilleur résultat). L'erreur $e(\Delta)$ est inférieure à 0.055 pour cinq caméras. L'erreur d augmente et $e(\Delta)$ diminue par les AFs non-centrals. La synchronisation à précision sous-trame raffinée sans AF (Section I.4) donne le plus petit $e(\Delta)$.

Méthode appliquée à BC2	$e(\Delta)$	$e(\tau)$	d
gs.c.fa.int+rs.c.sfa	0.057	14.6%	1.970
rs.c.sfa.int	0.097	2.7%	1.476
gs.nc.fa.int+rs.nc.sfa	0.051	12.2%	1.312
rs.nc.sfa.int	0.111	3.7%	0.366
gs.sfa (dans Sec.I.4)	0.215	na	na
Méthode appliquée à CC	$e(\Delta)$	$v_{max}f\tau$	d
gs.c.fa.int+rs.c.sfa	0.052	-0.0052	1.176
rs.c.sfa.int	0.055	-0.0069	1.167
gs.nc.fa.int+rs.nc.sfa	0.034	-0.0020	1.322
rs.nc.sfa.int	0.039	0.0086	1.313
gs.sfa (dans Sec.I.4)	6e-3	na	na

TABLE I.3: Précision d'AFs rs.sfa.X(int) pour BC2 et pour CC. Rappel : d est convertie en pixels. "na" (indisponible) pour les données manquantes.

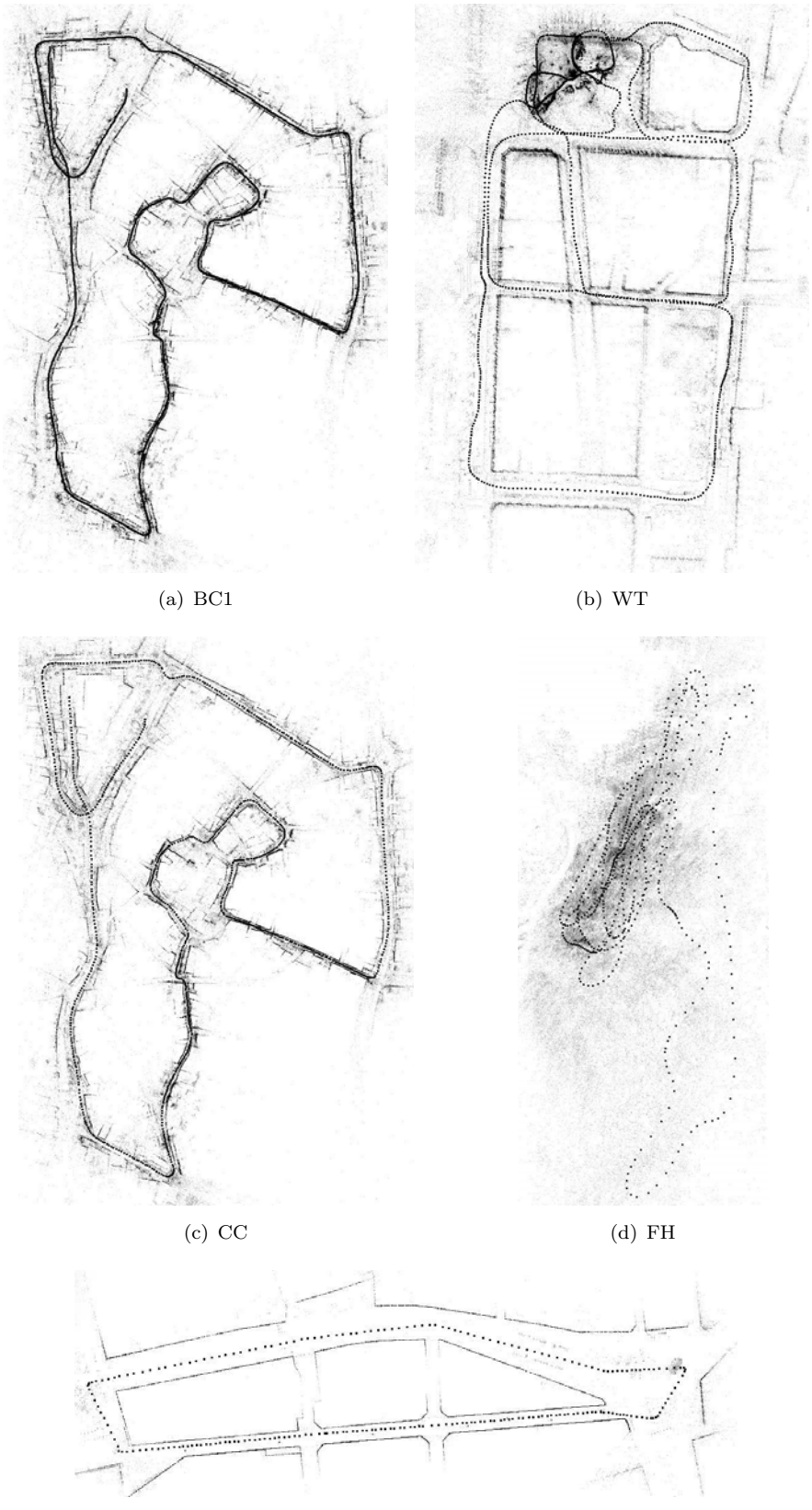
I.7.7.2 Décalages sous-trames, coefficient de RS et reconstruction

La Table I.4 montre les décalages sous-trames, le coefficient normalisé de RS $v_{\max}f\tau$ et l'erreur relative $e(\tau)$ pour tous les jeux de données par notre méthode avec l'approximation de calibration centrale et l'approximation de trajectoire de caméra donnée par l'Eq.(I.13). L'erreur $e(\tau)$ est inférieure à 7.2% sauf le cas de WT. Dans le cas de WT, τ est surestimé (il est même supérieur à sa valeur maximale théorique 1) et a une grande erreur égale à 16%. Une valeur négative de τ (pour WU) signifie simplement que le rolling shutter va de bas en haut dans les images (plutôt que de haut en bas).

Nom	$f\Delta_1$	$f\Delta_2$	$f\Delta_3$	$v_{\max}f\tau$	VT	$e(\tau)$
BC1	-0.334	-0.153	0.132	0.8755	0.8736	0.2%
WT	-0.583	-0.320	-0.795	1.0136	0.8736	16.0%
FH	0.287	0.203	-0.326	0.8372	0.7810	7.2%
BC2	0.246	0.546	0.797	0.8989	0.8755	2.7%
CC	-0.017	-0.013	-0.006	-0.0069	0	nan
WU	0.001	na	na	-0.8882	0.9244	3.9%

TABLE I.4: Décalages sous-trames entre caméras estimés et coefficient τ de RS (pour tous les jeux de données) estimés par notre AF raffinant simultanément la synchronisation, le rolling shutter, la calibration centrale (extrinsèque et intrinsèque), les poses et les points 3D. VT est la vérité terrain de $v_{\max}f\tau$.

Les Figures I.8 et I.9 illustrent les résultats de reconstruction après notre AF global (les poses de trames clefs et le nuage des points 3D). Pour le jeu de donnée WU, on observe une dérive non-négligeable de la trajectoire car le début et la fin de la trajectoire devraient être les mêmes (la dérive est moins notable dans les autres exemples). On explique cela de la façon suivante : (1) on n'applique pas de fermeture de boucles ; (2) le SfM multi-caméra incrémental est appliqué en utilisant la calibration intermédiaire estimée sur seulement 2k (premières) images ; et (3) l'AF final ne compense pas cette dérive. On refait le SfM multi-caméra incrémental en utilisant la calibration multi-caméra finale (estimée sur la séquence complète par RS.C.SFA.INT) et on voit que la dérive obtenue est plus faible. Ceci suggère que la calibration multi-caméra finale est meilleure que celle intermédiaire.



(a) BC1

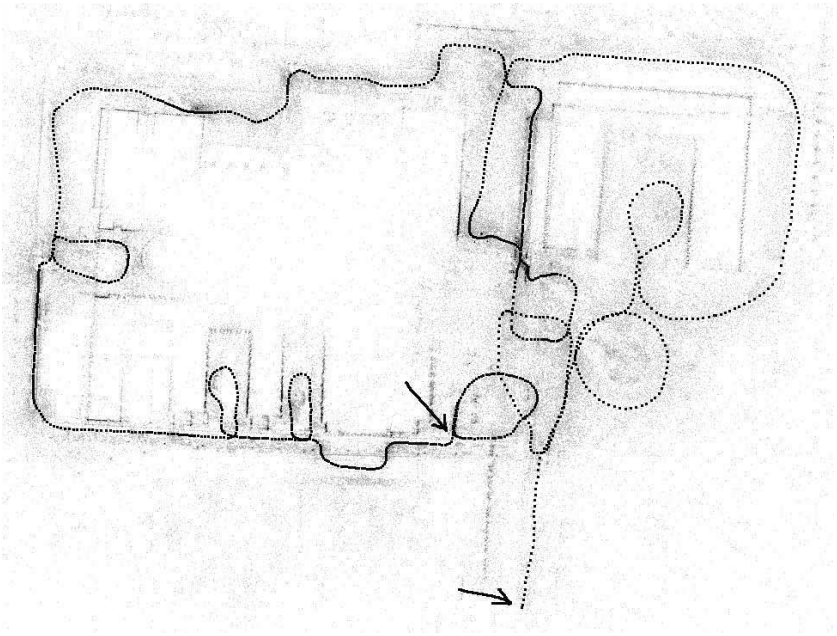
(b) WT

(c) CC

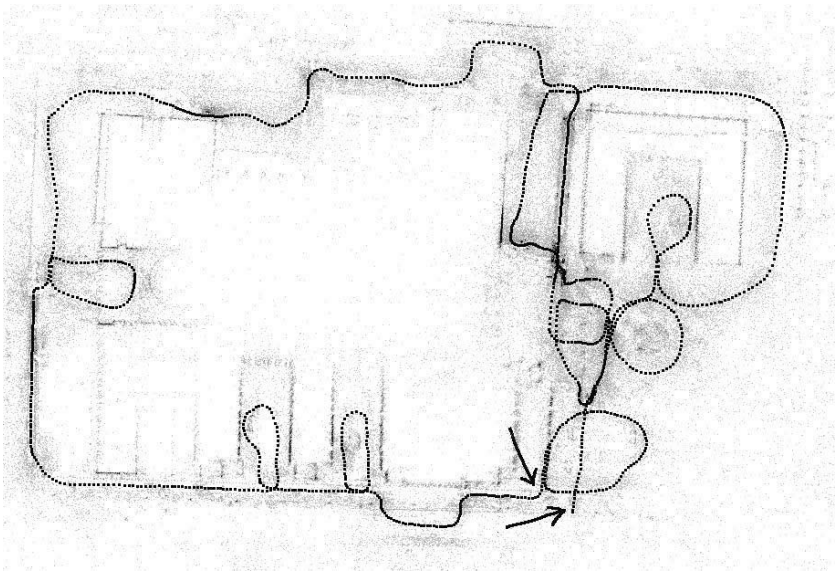
(d) FH

(e) BC2

FIGURE I.8: Vue globale du nuage de points 3D et des poses d'images clefs obtenus en utilisant l'AF RS.C.SFA.INT pour les séquences BC1, WT, FH, BC2 and CC sans fermeture de boucles. On note que la trajectoire de la séquence FH a beaucoup de virages.



(a) RS.C.SFA.INT



(b) SfM incrémental en utilisant la calibration estimée par l'AF RS.C.SFA.INT

FIGURE I.9: Vue globale du nuage de points 3D et des poses d'images clefs obtenus en utilisant l'AF RS.C.SFA.INT pour la séquence WU sans fermeture de boucles. La dérivée est entre les deux flèches

I.7.8 Autres expériences (résumé)

On expérimente la robustesse de notre méthode pour le cas de mauvaise synchronisation, mais aussi la stabilité de synchronisation et rolling shutter estimés par rapport aux sous échantillonnage en images clefs (le coefficient estimé de RS dépend de la distribution de trames clefs). Les trajectoires longues sont utiles pour la précision de l'étalonnage.

On vérifie l'approximation $t_{\mathbf{p}} = t_{\tilde{\mathbf{p}}}$ et le calcul exact de projection (Section I.6.3). On compare aussi les résultats entre les différents paramétrisations : les angles d'Euler et

quaternions pour rotation, l'approximation linéaire (Eq.(I.13)), quadratique et les interpolations sphériques pour la trajectoire de caméra. Les résultats montrent que l'on n'a pas de gain significatif si on utilise le calcul exact de projection ($t_{\mathbf{p}} \neq t_{\tilde{\mathbf{p}}}$) ou l'approximation quadratique ou les interpolation sphériques (*Squad* pour les quaternions unitaires [Dam+98] et spline [Jakubiak+06]) pour la trajectoire de caméra. Dans de nombreux cas (au moins dans le cas d'un déplacement à vélo ou à pied), l'approximation linéaire de trajectoire de caméra et l'approximation de projection suffisent. On teste aussi notre méthode dans le cas où le paramétrage d'Euler est proche de ses singularités.

On expérimente dans les cas où la vitesse de caméra varie dans les jeux de données synthétiques. Les résultats montrent que notre SfM standard fonctionne encore dans ces cas bien qu'il suppose que les caméras sont global shutter et synchronisées à précision trame. En comparaison avec les résultats de MCBA avec les approximations de global shutter et de synchronisation à précision trame, notre MCBA, y compris le RS et synchronisation à précision sous-trame, améliore significativement la précision en termes de l'erreur de calibration multi-caméra d.

I.8 Conclusion

Ce travail de thèse présente la première méthode d'auto-étalonnage pour un système multi-caméra se déplaçant dans une scène statique qui estime simultanément les paramètres intrinsèques, les poses inter-caméra, les décalages temporels et le coefficient de rolling shutter en plus des paramètres habituels (les poses multi-caméra et les points 3D). On commence par une calibration approximative en supposant que la multi-caméra est centrale et omnidirectionnelle sans direction privilégiée. Ensuite, on estime les décalages à précision trame en utilisant un structure-from-motion monoculaire classique et un ajustement de faisceaux (SfM et AF) sans faire l'hypothèse sur les champs de vue communs entre les caméras voisines. Enfin, on applique le SfM et l'AF multi-caméras deux fois : en utilisant les modèles simple et complexe de caméra. Contrairement au second, le premier modèle force à zéro le coefficient de rolling shutter, les décalages sous-trame entre les vidéos et la baseline entre caméras (le premier sert à initialiser le second).

On expérimente dans un contexte que nous pensons utile pour les applications (vidéo 360 et modélisation 3D) : plusieurs caméras grand public ou sphériques fixées sur un casque qui se déplacent des trajectoires longues à pied, vélo, ou autres. Les trajectoires longues sont utiles pour la précision de l'étalonnage, et sont permises car notre AF ne raffine que les images clés fournies par SfM. On compare les résultats central et non-central, donne la précision pour la calibration/les décalages entre caméras/le coefficient de rolling shutter par rapport à la vérité terrain, examine l'influence de sous-échantillonnage en images clefs, montre les variations de décalages entre caméras dans une séquence longue, expérimente et compare des approximations différentes pour la trajectoire de caméra et l'erreur de reprojection.

Des travaux futurs sont possibles pour toutes les étapes. Tout d'abord, l'initialisation de paramètres intrinsèques et extrinsèques peut être améliorée grâce à des travaux antérieurs. Deuxièmement, un pré-traitement devrait sélectionner un (des) segment(s) dans la vidéo pour appliquer le SfM en toute sécurité (les images doivent avoir suffisamment de texture, le mouvement de caméra ne doit pas avoir de rotation pure, les artefacts de RS doivent être modérés). Troisièmement, des variantes de la méthode peuvent être

expérimentées : d'autres méthodes de sous-échantillonnage en images clefs ou d'autres modèles de caméra. Enfin, il faudrait examiner les améliorations apportées à des applications fournies par notre synchronisation à précision sous-trame et si on tient compte du "rolling shutter".

Bibliography

Author's publications

[Lhuillier+15] Maxime Lhuillier and Thanh-Tin Nguyen. "Synchronization and self-calibration for helmet-held consumer cameras, applications to immersive 3d modeling and 360 video". In: *International Conference on 3D Vision (3DV)*. Oct. 2015.

[Nguyen+16a] Thanh-Tin Nguyen and Maxime Lhuillier. "Adding synchronization and rolling shutter in multi-camera bundle adjustment". In: *British Machine Vision Conference (BMVC)*. 2016.

[Nguyen+16b] Thanh-Tin Nguyen and Maxime Lhuillier. "Synchronisation et auto-étalonnage de plusieurs caméras fixées sur un casque". In: *Proc. Congrés National sur la Reconnaissance de Formes et l'Intelligence Artificielle (RFIA)*. 2016.

[Nguyen+17] Thanh-Tin Nguyen and Maxime Lhuillier. "Self-calibration of omnidirectional multi-cameras including synchronization and rolling shutter". In: *Computer Vision and Image Understanding (CVIU)*, vol. 162, pp. 166–184, 2017.

Bibliography

[360rize] 360rize. *360 Video hardware solution*. https://360rize.com/.

[Agapito+01] L. Agapito, E. Hayman, and I. Reid. "Self-calibration of rotating and zooming cameras". In: *International Journal of Computer Vision (IJCV)*, vol. 45(2), pp. 107–127, 2001.

[Agarwal+] S. Agarwal, K. Mierle, et al. *Ceres Solver*. http://ceres-solver.org.

[AitAider+06] 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 Computer Vison (ECCV)*. Springer, 2006.

[AitAider+09] O. Ait-Aider and F. Berry. "Structure and kinematics triangulation with a rolling shutter stereo rig". In: *IEEE 12th International Conference on Computer Vision (ICCV)*. Sept. 2009.

[Albl+15] C. Albl, Z. Kukelova, and T. Pajdla. "R6P - Rolling shutter absolute camera pose". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[Albl+16a] C. Albl, Z. Kukelova, and T. Pajdla. "Rolling shutter absolute pose problem with known vertical direction". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[Albl+16b] C. Albl, A. Sugimoto, and T. Pajdla. "Degeneracies in rolling shutter SfM". In: *European Conference on Computer Vision (ECCV)*. 2016.

[Andreff+01] N. Andreff, R. Horaud, and B. Espiau. "Robot hand-eye calibration using structure-from-motion". In: *The International Journal of Robotics Research (IJRR)*, vol. 20(3), pp. 228–248, 2001.

[Baker+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)*. June 2010.

[Barreto06] J.-P. Barreto. "A unifying geometric representation for central projection systems". In: *Computer Vision and Image Understanding (CVIU)*, vol. 103(3), pp. 208–217, 2006.

[Barreto03] J.-P.-A. Barreto. "General central projection systems: Modeling, calibration and visual servoing". PhD thesis. University of Coimbra, 2003.

[Bouguet02] J.-Y. Bouguet. *Complete camera calibration toolbox for Matlab®*. http://www.vision.caltech.edu/bouguetj/calib_doc/. 2002.

[Bradley+09] D. Bradley, B. Atcheson, I. Ihrke, and W. Heidrich. "Synchronization and rolling shutter compensation for consumer video camera arrays". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. June 2009.

[Bradski00] G. Bradski. *The OpenCV library*. Dr. Dobb's Journal of Software Tools, opencv.org. 2000.

[Brito+13] J.-H. Brito, R. Angst, K. Koser, and M. Pollefeys. "Radial distortion self-calibration". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013.

[Cannelle+12] B. Cannelle, N. Paparoditis, M. Pierrot-Deseilligny, and J.-P. Papelard. "Off-line vs. on-line calibration of a panoramic-based mobile mapping system". In: *XXII Congres ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. 2012.

[Cannelle+10] B. Cannelle, N. Paparoditis, and O. Tournaire. "Panorama-based camera calibration". In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science*. 2010.

[Carceroni+04] R.L. Carceroni, F.L.C. Padua, G.A.M.R. Santos, and K.N. Kutulakos. "Linear sequence-to-sequence alignment". In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2004.

[Carrera+11] G. Carrera, A. Angeli, and A.J. Davison. "SLAM-based automatic extrinsic calibration of a multi-camera rig". In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 2011.

[Caspi+00] Y. Caspi and M. Irani. "A step towards sequence-to-sequence alignment". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2000.

[Caspi+02a] Y. Caspi and M. Irani. "Aligning non-overlapping sequences". In: *International Journal of Computer Vision*, vol. 48(1), pp. 39–51, 2002.

[Caspi+02b] Y. Caspi and M. Irani. "Spatio-temporal alignment of sequences". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24(11), pp. 1409–1424, Nov. 2002.

[Caspi+06] Y. Caspi, D. Simakov, and M. Irani. "Feature-based sequence-to-sequence matching". In: *International Journal of Computer Vision (IJCV)*, vol. 68(1), pp. 53–64, 2006.

[Dai+16] Y. Dai, H. Li, and L. Kneip. "Rolling shutter camera relative pose: generalized epipolar geometry". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[Dai+09] Y. Dai, J. Trumpf, H. Li, N. Barnes, and R. Hartley. "Rotation averaging with application to camera-rig calibration". In: *Asian Conference on Computer Vision*. Springer, 2009.

[Dam+98] E.-B. Dam, M. Koch, and M. Lillholm. *Quaternions, interpolation and animation*. Technique report DIKU-TR-98/5. Department of Computer Science, University of Copenhagen, 1998.

[Devernay07] F. Devernay. *C/C++ Minpack*. http://devernay.free.fr/hacks/cminpack/. 2007.

[Duchamp+15] G. Duchamp, O. Ait-Aider, E. Royer, and J.-M. Lavest. "Multiple view 3D reconstruction with rolling shutter cameras". In: *10th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*, pp. 227–239. 2015.

[Esquivel+07] S. Esquivel, F. Woelk, and R. Koch. "Calibration of a multi-camera rig from non-overlapping views". In: *Joint Pattern Recognition Symposium*. Springer, 2007.

[Fassi+05] I. Fassi and G. Legnani. "Hand to sensor calibration: A geometrical interpretation of the matrix equation AX= XB". In: *Journal of Field Robotics*, vol. 22(9), pp. 497–506, 2005.

[Faugeras93] O. Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. Massachusetts Institute of Technology (MIT) press, 1993.

[Fischler+81] M.-A. Fischler and R.-C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the Association for Computing Machinery (ACM)*, vol. 24(6), pp. 381–395, 1981.

[Fitzgibbon01] A.-W. Fitzgibbon. "Simultaneous linear estimation of multiple view geometry and lens distortion". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 2001.

[Forssen+10] P.-E. Forssen and E. Ringaby. "Rectifying rolling shutter video from hand-held devices". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2010.

[Furgale+13] P. Furgale, J. Rehder, and R. Siegwart. "Unified temporal and spatial calibration for multi-sensor systems". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Nov. 2013.

[Gallier11] J. Gallier. *Geometric methods and applications: for computer science and engineering*. Springer Science & Business Media, 2011.

[Gaspar+14] T. Gaspar, P. Oliveira, and P. Favaro. "Synchronization of two independently moving cameras without feature correspondences". In: *European Conference on Computer Vision (ECCV)*. 2014.

[Gear360] Gear360. *Gear 360 camera*. http://www.samsung.com/global/galaxy/gear-360/.

[Geyer+00] C. Geyer and K. Daniilidis. "A unifying theory for central panoramic systems and practical implications". In: *European Conference on Computer Vision (ECCV)*. Springer, 2000.

[GoPro] GoPro. *Gopro camera*. https://gopro.com/.

[Grundmann+12] M. Grundmann, V. Kwatra, D. Castro, and I. Essa. "Calibration-free rolling shutter removal". In: *IEEE International Conference on Computational Photography (ICCP)*. Apr. 2012.

[Guennebaud+10] G. Guennebaud, B. Jacob, et al. *Eigen v3*. http://eigen.tuxfamily.org. 2010.

[Habib+11] A. Habib, A. Kersting, K. Bang, and J. Rau. "A novel single-step procedure for the calibration of the mounting parameters of a multi-camera terrestrial mobile mapping system". In: *Archives of Photogrammetry, Cartography and Remote Sensing*, vol. 22, pp. 173–195, 2011.

[Haralick+94] B.-M. Haralick, C.-N. Lee, K. Ottenberg, and M. Nölle. "Review and analysis of solutions of the three point perspective pose estimation problem". In: *International Journal of Computer Vision (IJCV)*, vol. 13(3), pp. 331–356, 1994.

[Harris+88] C. Harris and M. Stephens. "A combined corner and edge detector." In: *Alvey Vision Conference*. 1988.

[Hartley+04] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Second edition. Cambridge University Press, 2004.

[Hartley+13] R. Hartley, J. Trumpf, Y. Dai, and H. Li. "Rotation averaging". In: *International Journal of Computer Vision (IJCV)*, vol. 103(3), pp. 267–305, 2013.

[Hedborg+12] J. Hedborg, P.-E. Forssen, M. Felsberg, and E. Ringaby. "Rolling shutter bundle adjustment". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2012.

[Hedborg+11] J. Hedborg, E. Ringaby, P.-E. Forssen, and M. Felsberg. "Structure and motion estimation from rolling shutter video". In: *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. Nov. 2011.

[Hemayed03] E.-E. Hemayed. "A survey of camera self-calibration". In: *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance*. 2003.

[Heng+14] L. Heng, M. Burki, G.-H. Lee, P. Furgale, R. Siegwart, and M. Pollefeys. "Infrastructure-based calibration of a multi-camera rig". In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 2014.

[Heng+15] L. Heng, P. Furgale, and M. Pollefeys. "Leveraging image-based localization for infrastructure-based calibration of a multi-camera rig". In: *Journal of Field Robotics*, vol. 32(5), pp. 775–802, 2015.

[Heng+13] L. Heng, B. Li, and M. Pollefeys. "CamOdoCal: Automatic intrinsic and extrinsic calibration of a rig with multiple generic cameras and odometry". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Nov. 2013.

[Ikeda+03] S. Ikeda, T. Sato, and N. Yokoya. "Calibration method for an omnidirectional multicamera system". In: *Electronic Imaging*, pp. 499–507. 2003.

[Ikits00] M. Ikits. *Coregistration of pose measurement devices using nonlinear least squares parameter estimation*. Technical Report UUCS-00-018. School of Computing, University of Utah, 2000.

[Ito+17] E. Ito and T. Okatani. "Self-calibration-based approach to critical motion sequences of rolling-shutter structure from motion". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[Jakubiak+06] J. Jakubiak, F.-S. Leite, and R.-C. Rodrigues. "A two-step algorithm of smooth spline generation on Riemannian manifolds". In: *Journal of Computational and Applied Mathematics*, vol. 194(2), pp. 177–191, 2006.

[Kahl+02] F. Kahl and R. Hartley. "Critical curves and surfaces for Euclidean reconstruction". In: *European Conference on Computer Vision (ECCV)*, pp. 797–800. Springer, 2002.

[Kahl+99] F. Kahl and B. Triggs. "Critical motions in Euclidean structure from motion". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 1999.

[Karpenko+11] A. Karpenko, D. Jacobs, J. Baek, and M. Levoy. "Digital video stabilization and rolling shutter correction using gyroscopes". In: *Stanford University Computer Science Tech Report*, vol. 1, p. 2, 2011.

[Kim+06] J.-H. Kim and M.-J. Chung. "Absolute motion and structure from stereo image sequences without stereo correspondence and analysis of degenerate cases". In: *Pattern Recognition*, vol. 39(9), pp. 1649–1661, 2006.

[Kim+08] J.-S. Kim, M. Hwangbo, and T. Kanade. "Motion estimation using multiple non-overlapping cameras for small unmanned aerial vehicles". In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 2008.

[Klingner+13] B. Klingner, D. Martin, and J. Roseborough. "Street view motion-from-structure-from-motion". In: *The IEEE International Conference on Computer Vision (ICCV)*. Dec. 2013.

[Kumar+08] R.K. Kumar, A. Ilie, J.-M. Frahm, and M. Pollefeys. "Simple calibration of non-overlapping cameras with a mirror". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2008.

[Ladybug2] Ladybug2. *Ladybug2 camera*. https://eu.ptgrey.com/.

[Lamprecht+07] B. Lamprecht, S. Rass, S. Fuchs, and K. Kyamakya. "Extrinsic camera calibration for an on-board two-camera system without overlapping field of view". In: *IEEE on Intelligent Transportation Systems Conference (ITSC)*. Sept. 2007.

[Lavest+98] J.-M. Lavest, M. Viala, and M. Dhome. "Do we really need an accurate calibration pattern to achieve a reliable camera calibration?" In: *European Conference on Computer Vision (ECCV)*. Springer, 1998.

[Lébraly+10a] P. Lébraly, C. Deymier, O. Ait-Aider, E. Royer, and M. Dhome. "Flexible extrinsic calibration of non-overlapping cameras using a planar mirror: Application to vision-based robotics". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2010.

[Lébraly+11] P. Lébraly, E. Royer, O. Ait-Aider, C. Deymier, and M. Dhome. "Fast calibration of embedded non-overlapping cameras". In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 2011.

[Lébraly+10b] P. Lébraly, E. Royer, O. Ait-Aider, and M. Dhome. "Calibration of non-overlapping cameras - Application to vision-based robotics". In: *Proceedings of the British Machine Vision Conference (BMVC)*. 2010.

[Levenberg44] K. Levenberg. "A method for the solution of certain non-linear problems in least squares". In: *Quarterly of applied mathematics*, vol. 2(2), pp. 164–168, 1944.

[Lhuillier+13] M. Lhuillier and S. Yu. "Manifold surface reconstruction of an environment from sparse Structure-from-Motion data". In: *Computer Vision and Image Understanding*, vol. 117(11), pp. 1628–1644, 2013.

[Li+13] B. Li, L. Heng, K. Koser, and M. Pollefeys. "A multiple-camera system calibration toolbox using a feature descriptor-based calibration pattern". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Nov. 2013.

[Li+05] H. Li, R. Hartley, and L. Wang. "Auto-calibration of a compound-type omnidirectional camera". In: *Proceedings of Digital Image Computing: Techniques and Applications, (DICTA)*. Dec. 2005.

[Liang+08] C.-K. Liang, L.-W. Chang, and H.H. Chen. "Analysis and compensation of rolling shutter effect". In: *IEEE Transactions on Image Processing*, vol. 17(8), pp. 1323–1330, Aug. 2008.

[Litvinov15] V. Litvinov. "Incremental reconstruction of a complex scene using omnidirectional camera". PhD thesis. Université Blaise Pascal-Clermont-Ferrand II, 2015.

[Litvinov+14] V. Litvinov and M. Lhuillier. "Incremental solid modeling from sparse structure-from-motion data with improved visual artifacts removal". In: *International Conference on Pattern Recognition (ICPR)*, pp. 2745–2750. 2014.

[Lourakis+09] M.-I. Lourakis and A.-A. Argyros. "SBA: A software package for generic sparse bundle adjustment". In: *ACM Transactions on Mathematical Software (TOMS)*, vol. 36(1), p. 2, 2009.

[Lovegrove+13] S. Lovegrove, A. Patron-Perez, and G. Sibley. "Spline Fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras". In: *Proceedings of the British Machine Vision Conference (BMVC)*. 2013.

[Luong+93] Q.-T. Luong and O. Faugeras. *Self-calibration of a stereo rig from unknown camera motions and point correspondences*. Research Report RR-2014. INRIA, 1993.

[Madsen+04] K. Madsen, H.-B. Nielsen, and O. Tingleff. "Methods for non-linear least squares problems". In: *Lecture notes, Informatics and Mathematical Modelling, Technical University of Denmark*, pp. 1–60, 2004.

[Magerand+12] 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)*. Springer, 2012.

[Marquardt63] D.-W. Marquardt. "An algorithm for least-squares estimation of nonlinear parameters". In: *Journal of the society for Industrial and Applied Mathematics*, vol. 11(2), pp. 431–441, 1963.

[Matlab17] Matlab. *Optimization Toolbox*. The MathWorks, Natick, MA, USA. 2017.

[Meilland+13] 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)*. Dec. 2013.

[Meingast+05] L. Meingast, C. Geyer, and S. Sastry. "Geometric models of rolling-shutter cameras". In: *Omnidirectional Vision, Camera Networks and Non-classical Cameras (OMNIVIS)*. Oct. 2005.

[Meyer+08] B. Meyer, T. Stich, M.-A. Magnor, and M. Pollefeys. "Subframe temporal alignment of non-stationary cameras." In: *British Machine Vision Conference (BMVC)*. 2008.

[Micusik+06] B. Micusik and T. Pajdla. "Structure from motion with wide circular field of view cameras". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28(7), pp. 1135–1149, 2006.

[Mouragnon+07] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. "Generic and real-time structure from motion." In: *Bristish Machine Vison Conference (BMVC)*. 2007.

[Mouragnon+09] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. "Generic and real-time structure from motion using local bundle adjustment". In: *Image and Vision Computing*, vol. 27(8), pp. 1178–1193, 2009.

[Nielsen99] H.-B. Nielsen. *Damping parameter in Marquardt's method*. Technical report. Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 1999.

[Nistér04] D. Nistér. "An efficient solution to the five-point relative pose problem". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26(6), pp. 756–770, 2004.

[Nyman+10] P. Nyman, A. Heyden, and K. Astrom. "Multi-camera platform calibration using multi-linear constraints". In: *20th International Conference on Pattern Recognition (ICPR)*. Aug. 2010.

[Oskiper+07] T. Oskiper, Z. Zhu, S. Samarasekera, and R. Kumar. "Visual odometry system using multiple stereo cameras and inertial measurement unit". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2007.

[Oth+13] L. Oth, P. Furgale, L. Kneip, and R. Siegwart. "Rolling shutter cmera calibration". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2013.

[Park+94] F.-C. Park and B.-J. Martin. "Robot sensor calibration: solving AX= XB on the Euclidean group". In: *IEEE Transactions on Robotics and Automation*, vol. 10(5), pp. 717–721, 1994.

[PatronPerez+15] A. Patron-Perez, S. Lovegrove, and G. Sibley. "A spline-based trajectory representation for sensor fusion and rolling shutter cameras". In: *International Journal of Computer Vision*, vol. 113(3), pp. 208–219, 2015.

[Press+96] W.-H. Press, S.-A. Teukolsky, W.-T. Vetterling, and B.-P. Flannery. *Numerical recipes in C*. Second edition. Cambridge University Press, 1996.

[Ramadasan+17] D. Ramadasan, M. Chevaldonné, and T. Chateau. "LMA: A generic and efficient implementation of the Levenberg–Marquardt Algorithm". In: *Software: Practice and Experience* 2017.

[Rao+03] C. Rao, A. Gritai, M. Shah, and T. Syeda-Mahmood. "View-invariant alignment and matching of video sequences". In: *Proceedings of 9-th IEEE International Conference on Computer Vision (ICCV)*. Oct. 2003.

[Ringaby+11] E. Ringaby and P.-E. Forssén. "Efficient video rectification and stabilisation for cell-phones". In: *International Journal of Computer Vision (IJCV)*, vol. 96(3), pp. 335–352, 2011.

[Royer+07] E. Royer, M. Lhuillier, M. Dhome, and J.-M. Lavest. "Monocular vision for mobile robot localization and autonomous navigation". In: *International Journal of Computer Vision*, vol. 74(3), pp. 237–260, 2007.

[Sattinger+13] D.-H. Sattinger and O.-L. Weaver. *Lie groups and algebras with applications to physics, geometry, and mechanics*. Volume 61. Springer Science & Business Media, 2013.

[Saurer+13] O. Saurer, K. Koser, J.-Y. Bouguet, and M. Pollefeys. "Rolling shutter stereo". In: *IEEE International Conference on Computer Vision (ICCV)*. Dec. 2013.

[Saurer+15] O. Saurer, M. Pollefeys, and G.-H. Lee. "A minimal solution to the rolling shutter pose estimation problem". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2015.

[Saurer+16] O. Saurer, M. Pollefeys, and H.-G. Lee. "Sparse to dense 3D reconstruction from rolling shutter images". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[Schmidt+01] J. Schmidt and H. Niemann. "Using quaternions for parametrizing 3D rotations in unconstrained nonlinear optimization." In: *Vision, Modeling & Visualization (VMV)*. vol. 1, pp. 399–406. 2001.

[Schneider+13] J. Schneider and W. Förstner. *Bundle adjustment and system calibration with points at infinity for omnidirectional cameras*. Technical Report. Institute of Geodesy and Geoinformation, University of Bonn, 2013.

[Singla+04] P. Singla, D. Mortari, and J.-L. Junkins. "How to avoid singularity when using Euler angles". In: *Advances in the Astronautical Sciences Space Flight Mechanics Conference*, pp. 04–190. 2004.

[Solà+08] J. Solà, A. Monin, M. Devy, and T. Vidal-Calleja. "Fusing monocular information in multicamera SLAM". In: *IEEE Transactions on Robotics*, vol. 24(5), pp. 958–968, Oct. 2008.

[Spencer+04] L. Spencer and M. Shah. "Temporal synchronization from camera motion". In: *Proceedings of Asian Conference on Computer Vision (ACCV)*. 2004.

[Sturm97] P. Sturm. "Critical motion sequences for monocular self-calibration and uncalibrated Euclidean reconstruction". In: *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 1997.

[Sturm+11] P. Sturm, S. Ramalingam, J.-P. Tardif, S. Gasparini, and J. Barreto. "Camera models and fundamental concepts used in geometric computer vision". In: *Foundations and Trends® in Computer Graphics and Vision*, vol. 6(1–2), pp. 1–183, 2011.

[Thalin10] D. Thalin. *Deshaker. video stabilizer plugin, v2.5 for Virtual Dub*. http://guthspot.se/video/deshaker.htm. 2010.

[ThetaS] ThetaS. *Theta S camera*. https://theta360.com/.

[Tresadern+03] P.A. Tresadern and I. Reid. "Synchronizing image sequences of non-rigid objects." In: *British Machine Vision Conference (BMVC)*. 2003.

[Triggs+00] B. Triggs, P.-F. McLauchlan, R.I. Hartley, and A.-W. Fitzgibbon. "Bundle adjustment - a modern synthesis". In: *Vision algorithms: theory and practice*. Springer, 2000.

[Tsai86] R.-Y Tsai. "An efficient and accurate camera calibration technique for 3D machine vision". In: *Proceeding of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1986.

[Tuytelaars+04] T. Tuytelaars and L. Van Gool. "Synchronizing video sequences". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2004.

[Virb360] Virb360. *VIRB®360 camera*. https://buy.garmin.com/fr-FR/FR/p/562010/.

[Vo+16] M. Vo, S.-G. Narasimhan, and Y. Sheikh. "Spatiotemporal bundle adjustment for dynamic 3D reconstruction". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[Wang+12] P.-C. Wang, P.-C. Tsai, Y.-C. Chen, and Y.-H. Tseng. "One-step and two-step calibration of a portable panoramic image mapping system". In: *The XXII Congress of the International Society for Photogrammetry and Remote Sensing*. 2012.

[Wedge+06] D. Wedge, D. Huynh, and P. Kovesi. "Motion guided video sequence synchronization". In: *Asian Conference on Computer Vision (ACCV)*. Springer, 2006.

[Wilburn+04] B. Wilburn, N. Joshi, V. Vaish, M. Levoy, and M. Horowitz. "High-speed videography using a dense camera array". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2004.

[Wu14] C. Wu. "Critical configurations for radial distortion self-calibration". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2014.

[Yan+04] J. Yan and M. Pollefeys. "Video synchronization via space-time interest point distribution". In: *Advanced Concepts for Intelligent Vision Systems*, pp. 501–504. 2004.

[Zhang+15] M. Zhang, J. Yao, M. Xia, K. Li, Y. Zhang, and Y. Liu. "Line-based multi-label energy optimization for fisheye image rectification and calibration". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4137–4145. 2015.

[Zhang00] Z. Zhang. "A flexible new technique for camera calibration". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22(11), pp. 1330–1334, 2000.