POLITECNICO DI MILANO
Corso di Laurea magistrale in Ingegneria Informatica
Dipartimento di Elettronica e Informazione

Pose estimation of multi-stereo camera sensors with non overlapping fields of view

AIR Lab
Laboratorio di Intelligenza Artificiale
e Robotica del Politecnico di Milano

Relatore: Matteo Matteucci
Correlatore: Simone Mentasti

Tesi di Laurea di:
Emiliano Gagliardi, matricola 878539

Anno Accademico 2017-2018

A Cinzia, Annamaria, Angelo e Francesco

Abstract

In this document we explore the use of multiple rigidly coupled stereo cameras for self-localization in robotics applications. Sensors are arranged in such a way their fields of view do not overlap, resulting in a configuration that maximizes the view covering of the robot surrounding region, while allowing to perform its 3D reconstruction from a single perception. This placement choice complicates the calibration problem, that is solved by extending a state of the art multi-camera calibration solution to multi-stereo settings. Different relative pose estimation algorithms are proposed, either based on the 3D3D approach or the 3D2D approach, and cameras are used both in coupled and decoupled fashion. Tests on a real sensor show that the coupled approaches outperform decoupled ones. Multi-stereo reconstruction, matching and relative pose estimation procedures are developed in stereo camera like fashion, allowing for their reuse in stereo localization pipelines either performing visual odometry or visual SLAM. As a proof of concept, a simple visual odometry procedure is developed and run on a system of 4 stereo cameras, which is tracked by an external motion capture system for ground truth comparison.

Sommario

La capacità di un robot di navigate autonomamente nell'ambiente in cui è immerso è fondamentale per lo svoglimento di mansioni complesse come il monitoraggio, la sorveglianza, il trasporto e altri. A tale scopo è necessario che il robot percepisca l'ambiente, si localizzi in esso, pianifichi una traiettoria priva di collisioni, ed infine attui i motori in modo tale da effettuarla. Questa tesi è concentrata sul problema di localizzazione, in particolare quando l'ambiente circostante viene percepito tramite l'uso di camere. Tali sensori sono diventati popolari in applicazioni di robotica per il basso peso, lo scarso utilizzo di energia e la grande quantità di dati che sono in grado di raccogliere.

Oltre a camere singole, configurazioni differenti sono state utilizzate come camere stereo e camere ad ampio campo visivo. La configurazione stereo consiste in una coppia di camere rigidamente connesse e posizionate in modo tale da osservare la stessa scena, e sono ampiamente utilizzate perchè permettono di ricostruire in 3D la scena osservata da una singola percezione. Le camere ad ampio raggio visivo sono invece in grado di catturare una vasta porzione della scena circostante, così da incrementare la probabilità di osservare un buon numero di punti salienti, ossia riconoscibili da diversi punti di vista, che sono utili a risolvere il problema di localizzazione. Un altro valido approccio al fine di aumentare il campo visivo è quello di utilizzare più camere rigidamente connesse, in modo tale che il loro campo visivo non sia sovrapposto.

La configurazione presa in considerazione in questa tesi consiste in un insieme di multiple camere stereo posizionate in modo tale che il loro campo visivo non sia sovrapposto. Come conseguenza, camere appartenenti alla stessa coppia osservano la stessa scena, ma elementi appartenenti a diverse coppie hanno campo visivo disgiunto. Un simile sistema di camere porta vantaggi sia dalla configurazione stereo che da quella composta da camere multiple, permettendo così di ricostruire in 3D un' ampia regione circostante il robot.

In questa tesi vengono sviluppati gli step fondamentali di un algoritmo di localizzazione per un sistema composto da multiple camere stereo con campo visivo non sovrapposto. Questi consistono in ricostruzione, associazione di punti appartenenti alla scena in diverse osservazioni, e differenti soluzioni al problema di stima della posizione relativa date due percezioni. Le procedure sono sviluppate in modo tale da disporre delle stesse interfacce delle corrispettive per camera stereo, in modo tale da poter essere inserite in esistenti algoritmi di localizzazione per camere stereo. Siccome la configurazione del sensore è assunta nota, viene sviluppata una procedura di calibrazione, che sfrutta la condivisione del campo visivo di elementi appartenenti alla stessa coppia per aumentare il numero di vincoli sulla soluzione.

Differenti algoritmi di stima della posizione relativa sono confrontati utlizzando un sensore reale. Come prova del concetto di utilizzare un sistema di multiple stereo camere in una procedura di localizzazione per camera stereo, un algoritmo di odometria visuale basato sugli algoritmi proposti viene sviluppato e testato. I risultati sono confrontati con dati ad alta precisione ottenuti tramite un sistema esterno di tracciamento del moto.

Ringraziamenti

Un enorme ringraziamento va a mia madre Cinza e mio padre Angelo che hanno soddisfatto la condizione necessaria perché io compissi i miei anni di studio, ossia il sostegno economico. Con la sicurezza che, seppur fondamentale, questi considereranno il sostegno economico come un dovere che non richiede ringraziamento, devo comunque a loro, a mio fratello Francesco e alla mia fidanzata Annamaria un costante sostegno morale.
In secondo luogo desidero ringraziare il prof. Matteo Matteucci e il Dott. Simone Mentasti che in questi mesi mi hanno dato tutto il supporto necessario per riuscire a concludere la stesura di questa tesi.
Ringrazio i miei colleghi di studio, tra cui Emanuele, Leonardo, Riccardo, Guglio, Giulio, Edoardo e Samuele. Oltre che a simpatici svaghi, con loro ho condiviso lo stimolo ad un continuo automiglioramento.
Infine, non posso negare che le passioni esterne all'univeristà siano decisive nella riuscita di un percorso. Quindi ringrazio i formidabili pescatori e amici Giorgio, Jury, e Gabriele, nella speranza che questa splendida passione mantenga stretto il nostro legame a lungo.

Notation

$\mathbf{X}^w$	Three-dimensional point in reference system w
$\tilde{\mathbf{X}}^w$	Homogeneous representation of the three-dimensional point $\mathbf{X}^w$
$\mathbf{x}$	Image point
$\tilde{\mathbf{x}}$	Homogeneous representation of the image point $\mathbf{x}$
$\mathbf{R}_v^w$	Rotation matrix of reference system v expressed in reference system w
$\mathbf{t}_v^w$	Translation vector of reference system v expressed in reference system w
$\mathbf{T}_v^w$	Homogeneous transformation of reference system v expressed in reference system w
$\mathbf{K}$	Calibration matrix
$\mathbf{P}$	Projection matrix
π	Projection function mapping a three-dimensional point to its image
$\check{q}$	Quaternion
$\mathbf{r}$	Rodrigues rotation
$\cdot$	Dot product
$\times$	Cross product
$[\tilde{\mathbf{x}}]_\times$	Cross product matrix of $\tilde{\mathbf{x}}$

Contents

Chapter 1

Introduction

The ability of a mobile robot to autonomously navigate in its environment is crucial for the accomplishment of complex tasks as automated inspection, surveillance, transport and others. Succesfull navigation requires the robot to perceive the environment and localize itself in it, plan an obstacle free trajectory bringing to the goal, and finally to control motors in such a way to achieve the desired trajectory. The focus of this thesis is self-localization, in particular when camera sensors are employed for environment perception. This kind of sensors has gained popularity in robotics applications because they are extremely lightweight, consume little power and collect a large amount of data.

Beside single camera system, different configurations have been employed in the literature like stereo and wide field of view (FOV) cameras. Stereo sensors consist in a pair of cameras arranged in such a way they simultaneously observe the same scene, and are widely used since they allow to perform a reconstruction of the observed scene from a single perception. Wide FOV cameras give instead the advantage of capturing a big portion of the surrounding environment, incrementing the probability of observing good scene features recognizable from different views, useful for localization. A different approach to increase the size of the observed region is to arrange multiple cameras in such a way their FOVs do not overlap.

The camera configuration object of this thesis is a set of multiple stereo sensors arranged in such a way their FOVs do not overlap, meaning that each stereo sensor is composed of two cameras observing the same scene, but elements of two different stereo pairs have disjoint FOVs. This configuration brings advantages from both the stereo and multi-camera configurations, indeed it captures enough data to perform 3D reconstruction of a wide region

surrounding the robot.

1.1 Contributions

In this thesis we develop the core blocks of a camera localization pipeline consisting in scene reconstruction, environment features matching and different relative pose estimation solutions for a multi-stereo sensor with non overlapping FOVs. These are developed to provide the same interface of their corresponding for stereo sensors, allowing to apply existing stereo camera localization pipelines to a multi-stereo camera. Since the sensor configuration is assumed to be known, we develop calibration procedure for multi-stereo sensors, exploiting the FOV sharing of cameras belonging to the same stereo pair to increase the number of constraints on the solution.

Different relative pose estimation algorithms are experimentally compared on a real sensor. As a proof of concept of the use of a multi-stereo sensor in a stereo pipeline, we implement a minimal stereo visual odometry procedure and we run it on our multi-stereo sensor. Results are then compared with ground truth data given by an external motion capture system, tracking a visual marker placed on the multi-stereo sensor.

1.2 Thesis Outline

The structure of this thesis is organized as follows. In Chapter 2 we present background material, consisting in camera model, scene reconstruction, scene features extraction and matching, and relative pose estimation algorithms for a stereo sensor. In Chapter 3 we describe some state of the art localization pipelines for stereo cameras and multi-cameras, followed by literature solutions to the multi-camera calibration problem when different sensors FOVs do not overlap. In Chapter 4 we explain our localization pipeline building blocks, together with a minimal visual odometry procedure in which they can be employed. Finally we present our multi-stereo calibration procedure and we give details about implementation of the mentioned algorithms. In Chapter 5 we present calibration, relative pose estimation, and visual odometry experiments settings and results. Chapter 6 contains the conclusion, in which we briefly describe our work together with some possible extensions and work directions. Finally Appendix A contains background material, in particular different representations of a rigid body position employed in the developed algorithms.

Chapter 2

Theoretical Background

In this chapter we present the mathematical instruments that are used in the rest of the thesis. We start introducing the camera model in Section 2.1. In Section 2.2 we introduce stereo cameras and the advantages they offer, and in Sections 2.3 and 2.4 we explain some techniques for keypoints extraction and matching, fundamental tools exploited in all the algorithm presented in this thesis. In Section 2.5, we introduce three possible approaches to stereo camera localization, based on keypoints extraction. Finally, in section 2.6, we introduce some applications of the camera pose estimation problem.

2.1 Single Camera

Camera sensors are devices able to capture a 2D projection of the observed scene, by processing the light field emanating from this. The fundamental components of a camera are a lens and a set of photosensitive sensors, each one corresponding to an image pixel. The lens is responsible of the focus: ideally, the direction of the light rays spreading from a scene point is modified when the lens is met, in such a way they converge to a single photosensitive element. The incoming light intensity is measured by the sensors, and stored in the corresponding pixel. We now describe a mathematical model for cameras, and a practical way to retrieve its parameters known in the literature as camera calibration.

2.1.1 Camera Model

The behavior of real cameras is modeled by the so called *pinhole camera*. This ideal camera has no lens, and a single aperture point from which the light rays can pass. As result, among all light rays spreading from a scene

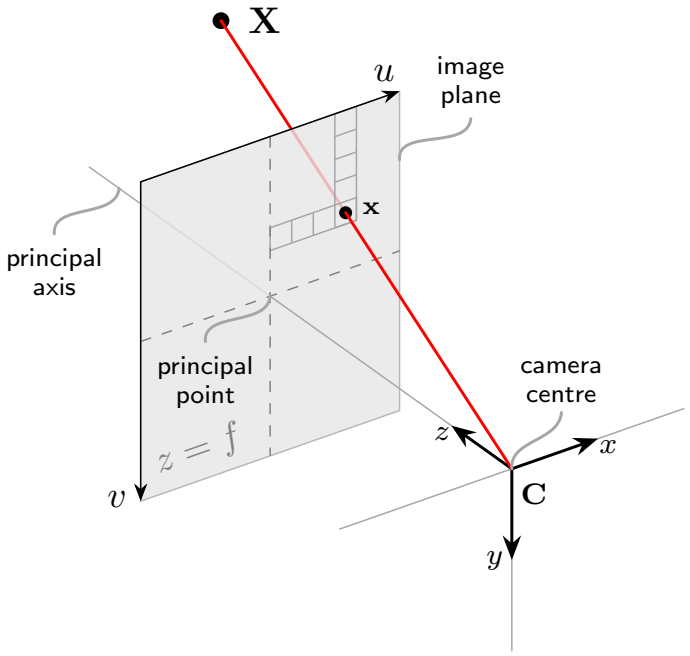


Figure 2.1: Central projection.

point, only the one passing through the image aperture will reach the plane on which the image is formed. All the light rays participating in the image formation intersect in the camera aperture, and the obtained image is a *central projection* of the observed scene.

A geometrical representation of the central projection is shown in Figure 2.1. A light ray, starting from the scene point $\mathbf{X}$, arrives to the projection center $\mathbf{C}$, called *camera center*, and intersects the *image plane*. On this, the image point $\mathbf{x}$ is formed. The distance f from the camera center to the image plane is called *focal distance*. The line through the camera center perpendicular to the image plane is called *principal axis*, and meets the image plane at the *principal point*. Conventionally, the camera reference frame is situated in the projection center, with the z axis placed on the principal axis.

Starting from this geometric model, we can derive an algebraic representation of the pinhole camera, as a mapping from 3D world points to 2D image points. Consider the triangle formed by the image point $\mathbf{x}$, the principal point, and $\mathbf{C}$, and the triangle formed by $\mathbf{X}$, $\mathbf{C}$, and the projection of $\mathbf{X}$

on the principal axis. By exploiting their similarity we obtain the relation:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{f}{Z} \begin{pmatrix} X \\ Y \end{pmatrix}, \tag{2.1}$$

where $\mathbf{x} = (x, y, f)$ and $\mathbf{X}^c = (X, Y, Z)^T$ is the expression of $\mathbf{X}$ in the camera reference frame. Let's now represent points as homogeneous vectors. Conveniently, the previous relation becomes a linear mapping, with the following expression:

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} = \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix}. \tag{2.2}$$

The next step is to represent image points as 2D homogeneous vectors on the image plane, with pixel coordinates u, v. Suppose that the principal point pixel coordinates are p_u and p_v, and that m_u and m_v are the number of pixels per distance unit in the u and v directions. By setting $f_u = m_u f$ and $f_v = m_v f$, the homogeneous expression of the image point $\mathbf{x}$ in pixel coordinates becomes:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{bmatrix} f_u & 0 & p_u & 0 \\ 0 & f_v & p_v & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix}, \tag{2.3}$$

that can be compactly written as $\tilde{\mathbf{x}} = \mathbf{K}[\mathbf{I} \mid \mathbf{0}]\tilde{\mathbf{X}}^c$, where $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{x}}$ are respectively the homogeneous representations of $\mathbf{x}$ and $\mathbf{X}$, $\mathbf{I}$ is the 3×3 identity matrix and $\mathbf{0}$ is the three-dimensional null vector. The matrix

$$\mathbf{K} = \begin{pmatrix} f_u & 0 & p_u \\ 0 & f_v & p_v \\ 0 & 0 & 1 \end{pmatrix} \tag{2.4}$$

is called *calibration matrix*, and its parameters are called *intrinsics*, because they only depends on the camera internal configuration.

Finally, consider the situation in which the coordinates of the world point are expressed in a different reference system from the one of the camera, that we call world frame:

$$\tilde{\mathbf{X}}^c = \begin{bmatrix} \mathbf{R}_w^c & \mathbf{t}_w^c \\ \mathbf{0}^T & 1 \end{bmatrix} \tilde{\mathbf{X}}^w. \tag{2.5}$$

The projective mapping can be written as $\tilde{\mathbf{x}} = \mathbf{P}\tilde{\mathbf{X}}^w$, where $\mathbf{P} = \mathbf{K}[\mathbf{R}_w^c \mid \mathbf{t}_w^c]$ is the *projection matrix*. The parameters $\mathbf{R}_w^c$ and $\mathbf{t}_w^c$ are called *extrinsics*,

since they do not depends on the camera internal configuration but only on its position in the world frame. We call π the projection function, mapping a 3D point to its image:

$$\mathbf{x} = \pi\left(\mathbf{K}, \mathbf{R}, \mathbf{t}, \mathbf{X}\right), \tag{2.6}$$

where $\mathbf{X}$ is a world point, $\mathbf{x}$ its image, and $\mathbf{K}$, $\mathbf{R}$, $\mathbf{t}$ the camera parameters.

2.1.2 Distortion

The pinhole model can be thought as a linear approximation of real cameras. Indeed, imperfections in lens manufacturing cause the presence of nonlinear effects in the world to image mapping, called *distortion*.

Let us consider three world points on a line, and their perspective projections. Since the three outgoing rays converges at the camera center, they lie on a single plane. By construction, the three corresponding image points must belong both to this plane and to the image plane, so to their intersection line. In this sense we say that perspective projection preserves collinearity. However, this property does not hold when a distortion effect is present, and it is not possible in general to apply the pinhole model without taking into account it. The most common kinds of distortion effect are *radial* and *tangential*, that we now briefly explain.

Radial distortion is mainly due to flawed radial lens curvature, and causes a radial displacement of points with respect to the principal point. The points coordinates aberration is modeled as:

$$\begin{aligned} u_d &= u(1 + k_1 r^2 + k_2 r^4 + k_3 r^6), \\ v_d &= v(1 + k_1 r^2 + k_2 r^4 + k_3 r^6), \end{aligned} \tag{2.7}$$

where points are represented in normalized image coordinates, meaning that they are centered in the principal point and divided by the focal distance expressed in pixels. Normalized image coordinates can be obtained by premultiplying an image point by the inverse calibration matrix. The values u_d and v_d represent distorted coordinates of the point ideally placed at (u, v), and $r = u^2 + v^2$ is the squared distance from the principal point. The values k_1, k_2, and k_3 are called *radial distortion parameters*.

Tangential distortion depends instead on errors in lens placement. In particular, it arises when the lens is not parallel to the image plane. We report here the mathematical model:

$$\begin{aligned} u_d &= u + (2p_1 uv + p_2(r^2 + 2u^2)), \\ v_d &= v + (p_1(r^2 + 2v^2) + 2p_2 uv), \end{aligned} \tag{2.8}$$

where p_1 and p_2 are the *tangential distortion parameters*.

The distortion parameters are considered intrinsics, because as focal distance and principal point position they describe the internal configuration of the camera. If all the distortion parameters are known, it is possible to correct the image deformation by computing for each pixel its ideal position on the image plane. This software correction is called *undistortion*, and makes possible to apply the pinhole model to real cameras with negligible error.

2.1.3 Calibration

Camera calibration is the process of estimating the intrinsic parameters of a camera, given some images acquired with it. Calibration algorithms can be divided in two categories: photogrammetric calibration and self-calibration. The first kind of algorithms requires to observe a known object, while the second requires only the camera to be moved in a static scene, and exploits its rigidity to obtain constraints on the intrinsic parameters. However, self-calibration gives in general poorer results with respect to the photogrammetric approach, and is not considered here.

The tools used in this thesis for calibration are based on [1], and use a planar checkerboard pattern. The use of this kind of pattern is convenient because the checkerboard corners are detected in the images with good sub-pixel accuracy. Since the pattern is known, it is possible to fit an homography $\mathbf{H}$, that maps pattern points to their corresponding images. This is done by expressing the pattern points in a frame placed on it, with coordinates $\tilde{\mathbf{X}} = (X, Y, 0, 1)$. The projective relation is simplified as:

$$\tilde{\mathbf{x}} = \mathbf{K}\,[\mathbf{R} \mid \mathbf{t}] \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} = \mathbf{K}\,[\mathbf{r}_1\,\mathbf{r}_2\,\mathbf{t}] \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}, \tag{2.9}$$

where $\mathbf{R} = [\mathbf{r}_1\,\mathbf{r}_2\,\mathbf{r}_3]$. The homography has expression $\mathbf{H} = \mathbf{K}\,[\mathbf{r}_1\,\mathbf{r}_2\,\mathbf{t}]$, and can be estimated in closed form if at least four correspondences are given. Note that if the calibration matrix is known, the extrinsic parameters can be recovered from $\mathbf{K}^{-1}\mathbf{H}$ by setting $\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2$.

If at least three images are provided, the calibration matrix parameters can be computed in closed form, imposing the following constraints for each homography:

$$\begin{aligned} &\mathbf{h}_1^T \mathbf{K}^{-T} \mathbf{K}^{-\mathbf{1}} \mathbf{h}_2 = 0, \\ &\mathbf{h}_1^T \mathbf{K}^{-T} \mathbf{K}^{-\mathbf{1}} \mathbf{h}_1 = \mathbf{h}_2^T \mathbf{K}^{-T} \mathbf{K}^{-\mathbf{1}} \mathbf{h}_2, \end{aligned} \tag{2.10}$$

where $\mathbf{H} = [\mathbf{h}_1 \mathbf{h}_2 \mathbf{h}_3]$.

It is now useful to introduce the notion of *reprojection error*, which represents the distance between a detected image points $\mathbf{x}$ and the projection on the image plane of its corresponding model point $\mathbf{X}$:

$$\mathbf{e} = \|\mathbf{x} - \pi(\mathbf{K}, \mathbf{R}, \mathbf{t}, \mathbf{X})\|. \tag{2.11}$$

After their estimation, camera matrix and extrinsic parameters are refined by minimizing the sum of squared reprojection errors over all the image points. This non-linear minimization problem is solved with the Levenberg-Marquardt optimization algorithm.

Distortion parameters are estimated after the refinement procedure, by comparing the detected image points with the corresponding reprojected points. Note that the expression of distorted points is linear in the distortion parameters, hence they can be computed in closed form. However, the intrinsic parameters estimation is performed over images that potentially presents distortion, employing the pinhole model. To solve this the calibration matrix and distortion parameters estimation procedures are alternated until convergence.

2.2 Stereo Camera

We consider a stereo camera as a system of two rigidly attached cameras whose FOVs have some overlapping region. From this kind of sensors two different images of the same scene can be captured simultaneously, giving the possibility of recovering the 3D position of an observed scene point by means of triangulation. A reconstruction pipeline needs to solve the stereo matching problem, namely finding corresponding points in the two views. Plus, cameras relative positioning and intrinsic parameters need to be known.

We start introducing epipolar geometry, an important tool for dealing with two views of the same scene. Then we describe stereo camera calibration, and image rectification: an image preprocessing procedure that allows to simplify the stereo matching problem. Finally we explain the triangulation process.

2.2.1 Epipolar Geometry

Epipolar geometry allows to define a relation between two images of the same scene point in two different views. This relation does not depend on the scene structure, but only on the intrinsic parameters and relative

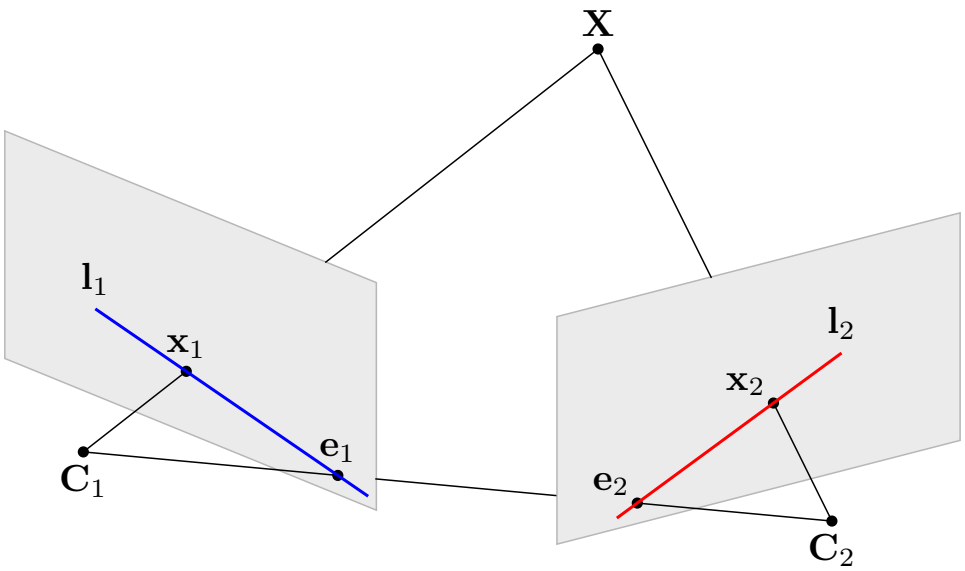


Figure 2.2: Epipolar geometry.

positioning of the cameras. Now we explain this relation by geometrical considerations.

Consider the situation shown in Figure 2.2, where a scene point $\mathbf{X}$ is observed by the two cameras with centers in $\mathbf{C}_1$ and $\mathbf{C}_2$, resulting in the two images $\mathbf{x}_1$ and $\mathbf{x}_2$. By construction, the two viewing rays lie on the same plane of the scene and image points, and same holds for the camera centers. This plane is called *epipolar plane*. The intersections between the image planes and the epipolar plane are called *epipolar lines*, while *epipolar points* are the intersections of the line through the camera centers, called *baseline*, and the image planes.

Now consider for instance the camera with center $\mathbf{C}_2$. The epipolar point $\mathbf{e}_2$ and the epipolar line $\mathbf{l}_2$ can be thought as image elements obtained by the second camera, since they lie on its image plane. In particular, $\mathbf{e}_2$ is the image of $\mathbf{C}_1$, and $\mathbf{l}_2$ is the image of the viewing ray through $\mathbf{x_1}$ and $\mathbf{X}$. The constraint expressed by epipolar geometry is that the image point $\mathbf{x}_2$ must belong to the epipolar line $\mathbf{l}_2$. Since the epipolar line is the image of the first camera viewing ray, we start finding an expression of it. This can be done by the notion of *back-projection ray*. We express the viewing ray line as a linear combination of two points belonging to it. In particular the two points are $\mathbf{C}_1$ and $\mathbf{P}_1^+\tilde{\mathbf{x}}_1$, where $\mathbf{P}_1^+ = \mathbf{P}_1^T\left(\mathbf{P}_1\mathbf{P}_1^T\right)^{-1}$ is the pseudo-inverse of the first camera projection matrix. Indeed we know that $\mathbf{P}_1\mathbf{P}_1^+\tilde{\mathbf{x}}_1 = \mathbf{I}\tilde{\mathbf{x}}_1 = \tilde{\mathbf{x}}_1$, so $\mathbf{P}_1^+\tilde{\mathbf{x}}_1$ must be on the same viewing ray of $\mathbf{x}_1$. Thus we obtain a set of

points parameterized by the scalar λ:

$$\mathbf{X}(\lambda) = \mathbf{P}_1^+ \tilde{\mathbf{x}}_1 + \lambda \mathbf{C}_1. \tag{2.12}$$

Now note that lines can be expressed in the geometry of a plane as homogeneous vectors. In particular, the line with expression $\mathbf{l} : ax + by + c$ is written as $\tilde{\mathbf{l}} = (a, b, c)^T$. The point $\mathbf{x}$ belongs to the line $\mathbf{l}$ if the dot product of their homogeneous expressions equals zero. Conveniently, the line passing through two points can be obtained by the cross product between their homogeneous representations.

For building the epipolar line $\mathbf{l}_2$ we choose the image of two points on the back-projection ray previously obtained, in particular with $\lambda = 0$ and $\lambda = \infty$. These are respectively $\mathbf{P}_1^+ \tilde{\mathbf{x}}_1$ and $\mathbf{C}_1$. Nothing that $\tilde{\mathbf{e}}_2 = \mathbf{P}_2 \tilde{\mathbf{C}}_1$, the expression of the epipolar line is:

$$\tilde{\mathbf{l}}_2 = [\tilde{\mathbf{e}}_2]_\times \mathbf{P}_2 \mathbf{P}_1^+ \tilde{\mathbf{x}}_1, \tag{2.13}$$

where $[\tilde{\mathbf{e}}_2]_\times$ is the skew symmetric matrix representing the cross product with $\tilde{\mathbf{e}}_2$:

$$[\tilde{\mathbf{e}}_2]_\times = \begin{bmatrix} 0 & -e_2^3 & e_2^2 \\ e_2^3 & 0 & -e_2^1 \\ -e_2^2 & e_2^1 & 0 \end{bmatrix}, \tag{2.14}$$

and $\tilde{\mathbf{e}}_2 = \left(e_2^1, e_2^2, e_2^3\right)$. The epipolar constraint has expression:

$$\tilde{\mathbf{x}}_2^T [\tilde{\mathbf{e}}_2]_\times \mathbf{P}_2 \mathbf{P}_1^+ \tilde{\mathbf{x}}_1 = 0, \tag{2.15}$$

where we imposed that the image point $\mathbf{x}_2$ belongs to the epipolar line $\mathbf{l}_2$ by setting their dot product equal to zero. The matrix

$$\mathbf{F} = [\tilde{\mathbf{e}}_2]_\times \mathbf{P}_2 \mathbf{P}_1^+ \tag{2.16}$$

is called *fundamental matrix*. By substituting it in the epipolar constraint we obtain the compact expression $\tilde{\mathbf{x}}_2^T \mathbf{F} \tilde{\mathbf{x}}_1 = 0$. This expression is important because it allows to compute the fundamental matrix if enough corresponding image points are given in the two views.

The matrix $\mathbf{F}$ is homogeneous, so only the ratios are important, leaving eight degrees of freedom. However, note that the fundamental matrix represents a mapping from points to lines: $\tilde{\mathbf{l}}_2 = \mathbf{F} \tilde{\mathbf{x}}_1$. All the image points of the first camera that are on the same epipolar plane as $\mathbf{x}_1$ are mapped to the same epipolar line in the second camera image plane. As a result $\mathbf{F}$ has not full rank, and so it has seven degrees of freedom. Thus seven points correspondences can be used to estimate it from two views.

An important aspect of the fundamental matrix is that it contains information about the relative positioning of the two cameras, up to a scale factor. This means that if the two cameras calibration matrices are known, it is possible to recover the rotation and the direction of the translation vector that relates the cameras frames.

2.2.2 Stereo Calibration

A stereo camera is considered to be calibrated when the intrinsic parameters of the forming cameras and their relative positioning are known. The stereo calibration process works as follows.

Suppose to take some stereo images of the planar pattern considered in Subsection 2.1.3 and to apply the single camera calibration procedure to both cameras. This gives as side effect the extrinsic parameters of both cameras for each captured image, and so the relative positioning of the two cameras. For a more robust solution, the relative positioning obtained from one image is taken as guess and refined minimizing the sum of squared reprojection errors over all the images using the Levenberg-Marquardt optimization algorithm.

2.2.3 Image rectification

Given a pair of stereo images, the rectification process transforms them in such a way that corresponding epipolar lines in the two views become collinear. After rectification the images can be thought as acquired by a new stereo rig, whose image planes are coplanar and pixels are row aligned. Note that in this situation the image planes are parallel to the baseline, and the epipoles are at the infinity. Indeed in each image all the epipolar lines are parallel, and meet at infinite distance in the epipole. For complete mathematical formulation of the constraints on the new camera configuration and image transformation refer to [2] and [3].

2.2.4 Triangulation

Suppose that two calibrated cameras observe the same scene, in particular the point $\mathbf{X}$ whose images are $\mathbf{x}_1$ and $\mathbf{x}_2$. If the two cameras relative position is known, it is possible to recover from the image points coordinates the 3D position of the point $\mathbf{X}$, as the intersection of the two viewing rays. However, due to noise in the measured image coordinates, the rays do to intersect in general. The solution is to find the point $\hat{\mathbf{X}}$ that minimizes the sum of squared reprojection error over the two images. Suppose without loss

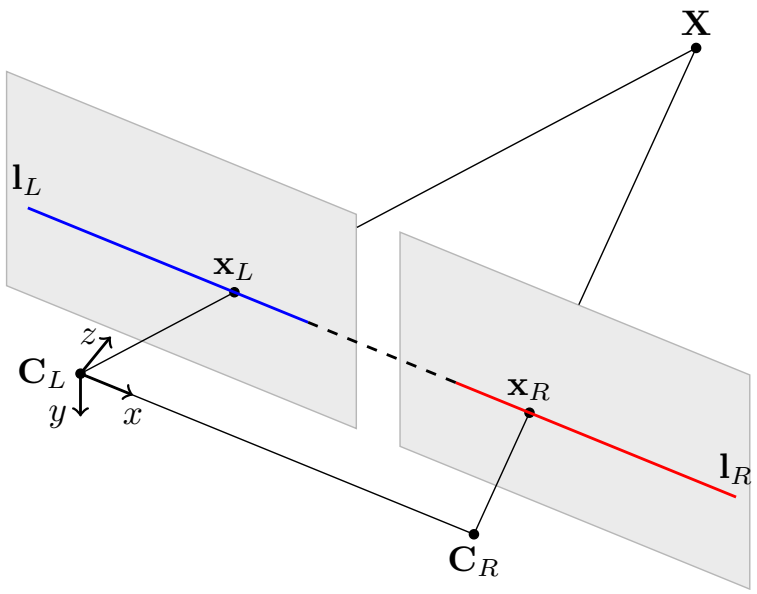


Figure 2.3: Rectified stereo pair.

of generality the two camera projection matrices to be respectively $\mathbf{P}_1 = \mathbf{K}_1 [\mathbf{I} \mid \mathbf{0}]$ and $\mathbf{P}_2 = \mathbf{K}_2 [\mathbf{R}_2 \mid \mathbf{t}_2]$, the problem has formulation:

$$\min_{\hat{\mathbf{X}}} \left(\left\| \mathbf{x}_1 - \pi \left(\mathbf{K}_1, \mathbf{I}, \mathbf{0}, \hat{\mathbf{X}} \right) \right\|^2 + \left\| \mathbf{x}_2 - \pi \left(\mathbf{K}_2, \mathbf{R}_2, \mathbf{t}_2, \hat{\mathbf{X}} \right) \right\|^2 \right). \tag{2.17}$$

However it might be inconvenient to solve such a problem, because it requires to find the roots of a polynomial of degree six [4]. In most practical implementation a different solution is computed, using the *linear triangulation method*. The method consists in solving the set of equations:

$$\begin{cases} \tilde{\mathbf{x}}_1 \times \mathbf{P}_1 \hat{\mathbf{X}} = \mathbf{0}, \\ \tilde{\mathbf{x}}_2 \times \mathbf{P}_2 \hat{\mathbf{X}} = \mathbf{0}; \end{cases} \tag{2.18}$$

where the cross product is used to eliminate the homogeneous factor. Indeed the equation $\lambda \tilde{\mathbf{x}} = \mathbf{P} \tilde{\mathbf{X}}$, for some scalar $\lambda \neq 0$, holds when the vectors at the two side of the equal sign have the same direction.

Consider the first equation of 2.18, this can be rewritten by exploiting the skew symmetric cross product operator:

$$[\mathbf{x}_1]_\times = \begin{bmatrix} 0 & -1 & v_1 \\ 1 & 0 & -u_1 \\ -v_1 & u_1 & 0 \end{bmatrix} \tag{2.19}$$

where $\mathbf{x}_1 = (u_1, v_1, 1)$. We obtain:

$$\begin{bmatrix} v_1\,\mathbf{p}_1^{3\,T} - \mathbf{p}_1^{2\,T} \\ u_1\,\mathbf{p}_1^{3\,T} - \mathbf{p}_1^{1\,T} \\ u_1\,\mathbf{p}_1^{2\,T} - v_1\,\mathbf{p}_1^{1\,T} \end{bmatrix} \hat{\mathbf{X}} = \mathbf{0}, \tag{2.20}$$

where $\mathbf{p}_1^{i\,T}$ is the i-th row of $\mathbf{P}_1$. By applying the same process on the point $\mathbf{x}_2$ we obtain a set of four linearly independent equations, with matrix form $\mathbf{A}\hat{\mathbf{X}} = \mathbf{0}$. This is an homogeneous linear equation system and can be solved by means of singular value decomposition of the matrix $\mathbf{A}$.

In the case of rectified images, the triangulation problem can be simply solved by triangle similarity [2]. Suppose that B_x is the baseline measured in number of horizontal pixels. We obtain:

$$\frac{B_x - (u_l - p_u) + (u_r - p'_u)}{B_x} = \frac{Z - f}{Z} \implies Z = \frac{f B_x}{d - (p_u - p'_u)}, \tag{2.21}$$

where u_l and u_r are the column coordinates of the left and right image points, p_u and p'_u the two cameras principal points column coordinates, and $d = u_l - u_r$ is called disparity. From the depth knowledge the other two coordinates are computed as:

$$X = \frac{Z}{f_x}(u - p_u), \tag{2.22}$$

$$Y = \frac{Z}{f_y}(v - p_v). \tag{2.23}$$

Now it is useful to inspect what is the effect of noise on the perceptions $\mathbf{x}_1$ and $\mathbf{x}_2$. Suppose to have radial uncertainty over the image points. The back-projection of the uncertainty circle around a point forms a cone, with apex in the camera center and the circle as directrix. The uncertainty zone of the reconstructed point lies in the intersection of the two back-projection cones. This shape strongly depends on the positioning of the cameras and the point.

However, when a stereo camera is employed, the classical situation is that the point to camera distance is way bigger than the baseline. As effect the depth estimation uncertainty is bigger than the other two coordinates. Plus notice that the uncertainty increase as the point depth increase. A two dimensional illustration of the phenomenon can be seen in Figure 2.4.

2.3 Image Features

Many geometrical computer vision algorithms are based on the assumption that corresponding points in two views of the same scene are known.

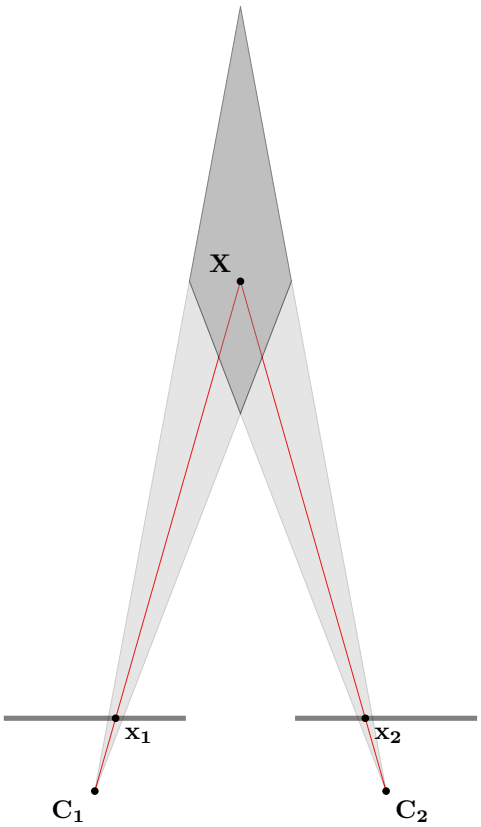


Figure 2.4: Uncertainty propagation in triangulation.

Two examples are the fundamental matrix estimation, that requires at least seven correspondences, and triangulation, that computes the 3D position of a point from two views of it. In general this task is accomplished in three steps. The first one is *feature extraction*, where each image is searched for locations that are likely to match well in other images, called *keypoints*. A good features extractor should be able to find the same interesting points under different viewing conditions. Then in the *feature description* stage, each keypoint is associated with a *descriptor vector*, computed from its surrounding region. The descriptor has to be distinctive and at the same time robust to changes of the view point. In the third stage descriptors from different images are matched by their similarity, in order to obtain keypoint features correspondences.

Two different kinds of detectors are mostly used in the literature: corners and blob detectors. A corner is defined as a point at the intersection of two or more edges, while a blob is an image patch that differs from its adjacent pixels.

2.3.1 Harris Corners Detector

Suppose to position a squared patch around an image point, and observe how the portion of image inside the patch changes as we move it. For untextured image parts we do not have changes in any direction, while for edges the patch content significantly changes only if we make a movement in the edge perpendicular direction. Instead, if we center the patch in a corner, significant changes are visible in any movement direction. The Harris corners detector [5] defines the energy function as the intensity change produced by the shift (u, v):

$$E(u,v) = \sum_{x,y} w(x,y) \left[I(x+u, y+v) - I(x,y) \right]^2 , \tag{2.24}$$

where $w(x, y)$ specifies the window, that could be unitary inside the patch and zero elsewhere, or a gaussian window as specified by the authors. The energy function computes the sum of squared distances between a image patch and its translation by (u, v).

Since we are looking for points with large energy for each (u, v) direction small displacement, it is sufficient to analyze the local behavior of the energy function. This is done by performing first-order Taylor expansion:

$$E(u,v) = \begin{bmatrix} u & v \end{bmatrix} \left(\sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix} = \mathbf{D}^T \mathbf{M} \mathbf{D}, \tag{2.25}$$

where I_x and I_y are image derivatives (generally computed using the Sobel operator) in x and y directions. The nature of the energy function can be deduced by the eigenvalues λ_1 and λ_2 of $\mathbf{M}$:

- If both are small, then the energy function is almost constant.
- If $\lambda_1 \gg \lambda_2$ or $\lambda_2 \gg \lambda_1$ the energy function increases in only one direction, thus we are dealing with a point belonging to an edge
- If both λ_1 and λ_2 are large, then E is increasing in all direction thus the (x, y) point is a corner.

To avoid eigenvalues computation, the relations $Tr(\mathbf{M}) = \lambda_1 + \lambda_2$ and $det(\mathbf{M}) = \lambda_1 \lambda_2$ are exploited in the response function:

$$R = det(\mathbf{M}) - kTr(\mathbf{M}), \tag{2.26}$$

that is big when both λ_1 and λ_2 are big.

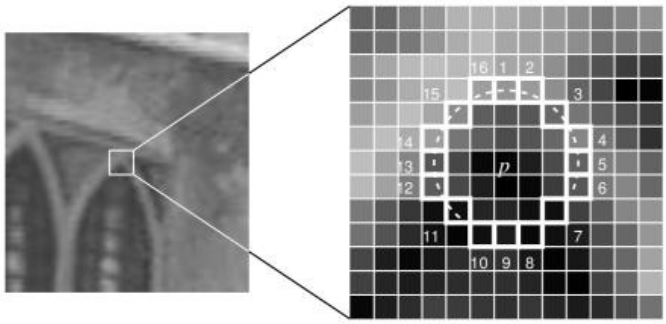


Figure 2.5: Full-segment test circle of FAST corner detector [6].

2.3.2 FAST Corners Detector

The FAST [6] (Features from Accelerated Segment Test) algorithm has become a popular alternative to the Harris corners detector because of its efficiency. For each target point p, a circle of 16 pixels around it is considered. In the full segment test a point p is classified as corner if there are at least n contiguous pixels in the circle with intensity higher than $I(p) + t$ or lower than $I(p) - t$, for a fixed threshold t. An high speed test was proposed to exclude non-corners before applying the full-segment test, by only considering the brightness of the pixels labeled as 1, 5, 9, 13 in figure 2.5. If at least three of the four points are brighter than $I(p) + t$ or darker than $I(p) - t$, then the initial test is passed.

2.3.3 BRIEF Descriptor

BRIEF [7] (Binary Robust Independent Elementary Features) is an algorithm that computes a binary descriptor vector of an image keypoint. This can be associated for instance with Harris or FAST features detectors.

Given an image patch $\mathbf{p}$ around a keypoint, the algorithm first performs Gaussian smoothing and then computes a binary descriptor by pairwise comparison of patch pixels at specific locations. In particular, each element of the binary string is computed as follows:

$$\tau(\mathbf{p}, \mathbf{x}, \mathbf{y}) = \begin{cases} 1 & \text{if } \mathbf{p}(\mathbf{x}) < \mathbf{p}(\mathbf{y}), \\ 0 & \text{otherwise,} \end{cases} \tag{2.27}$$

for some locations pair $(\mathbf{x}, \mathbf{y})$. The set of locations pairs defines the descriptor function. We show in Figure 2.6 some arrangement examples of the test locations:

I) Both $\mathbf{x}$ and $\mathbf{y}$ are uniformly distributed.

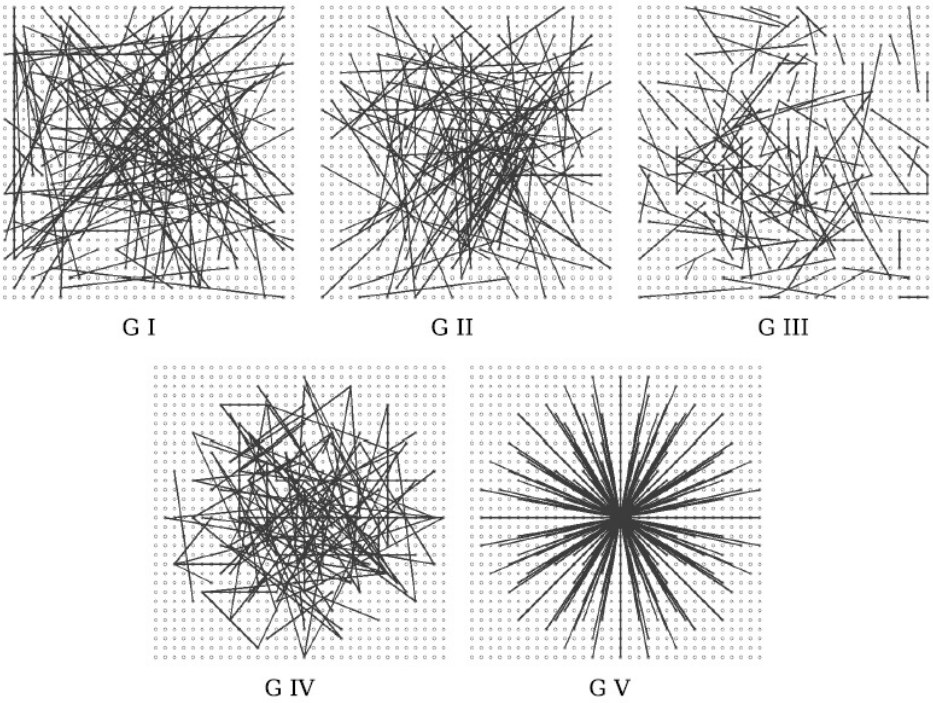


Figure 2.6: Different test locations arrangement for the BRIEF descriptor [7]. Coupled test locations are joined by a line.

II) Both $\mathbf{x}$ and $\mathbf{y}$ are distributed according to an isotropic Gaussian centered in the patch center.

III) $\mathbf{x}$ is sampled from a Gaussian distribution centered in the patch center, while $\mathbf{y}$ is sampled from a Gaussian distribution centered in the corresponding value of $\mathbf{x}$.

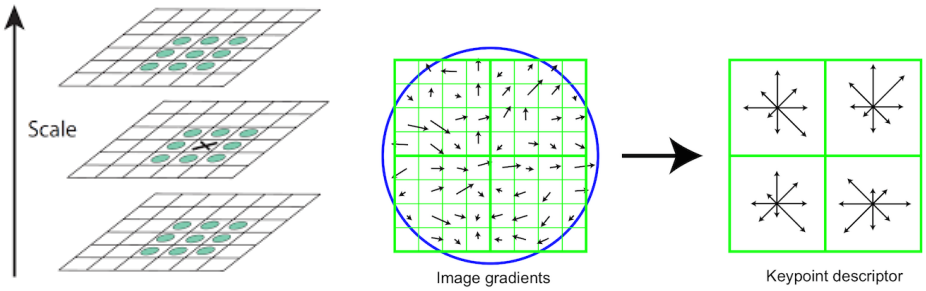
IV) Both $\mathbf{x}$ and $\mathbf{y}$ are randomly sampled on a coarse polar grid.

V) $\mathbf{x}$ is always located at the patch center, while $\mathbf{y}$ is sampled from a coarse polar grid.

Different test locations are compared by the authors over descriptors size of 128, 256 and 512 tests, with G I to G IV outperforming the regular arrangement of G V.

2.3.4 SIFT And SURF

SIFT [8] (Scale Invariant Features Transform) is a blob detector that also provides descriptors computation. It uses custom techniques for assuring



(a) Scale space local maxima detection in SIFT.

(b) Descriptor formation in SIFT.

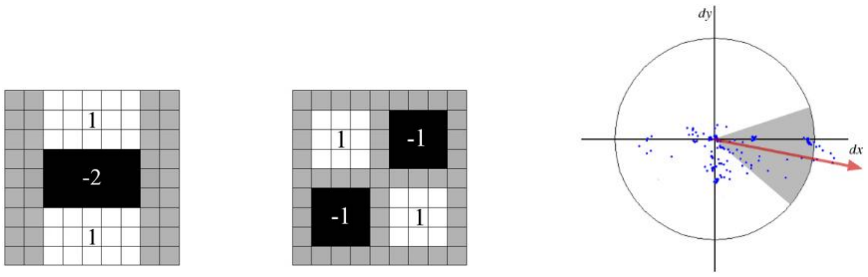
Figure 2.7: Scale invariant features transform [8].

rotation and scale invariance, and it is shown to provide robust matching against change in 3D viewpoint and light conditions.

To provide scale invariance, keypoints are searched in an image pyramid whose levels are views of the input image at different scales. Simulated images are obtained by Gaussian smoothing with higher standard deviation at higher levels and re-sampling. A keypoint is a local maximum in the difference between two adjacent levels in the scale space representation. The resulting operator takes the name of difference-of-Gaussians (DoG) because it consists in convolving the difference of two Gaussian kernels. Local maxima are obtained by comparison of adjacent DoG as shown in Figure 2.7a, and bad keypoints are removed by checking the eigenvalues of the DoG Hessian, similarly to the Harris corner detector. Each keypoint is associated with the scale at which it has been detected, information that is then used in the descriptor computation.

For orientation invariance, each keypoint is assigned with an angle computed from its 16×16 surrounding region at the selected scale. For each patch pixel the gradient is computed and an histogram of the gradients orientations is built. The peaks in the histogram provides the predominant orientation of local gradients. If more than one predominant orientation is present, more keypoints at the same location and scale are obtained, but with different orientations. This technique is called histogram-of-oriented-gradients (HOG).

The descriptor computation process is shown in Figure 2.7b. A keypoint descriptor is obtained from the image gradients in a 16×16 patch around it, at the corresponding scale space in the image pyramid. Gradients are then rotated according to the keypoint orientation, providing SIFT descriptor with rotation and scale invariance. As for orientation, the gradients angle



(a) Approximate second order gaussian derivatives. *(b) Orientation assignment in SURF.*

Figure 2.8: Speeded up robust features [9].

are weighted by their magnitude and by a Gaussian window centered in the keypoint. Then the patch is divided in 4×4 regions, each one with an 8-bin histogram. The histograms are stacked together forming a $4 \times 4 \times 8 = 128$ elements descriptor vector. Note that since gradients are used, descriptors are invariant to additive light changes. To also achieve invariance to intensity light scaling, descriptors are normalized, and better behavior in case of non-linear light changes is obtained by clipping descriptor elements before normalization.

A faster yet comparable in performance algorithm is SURF [9] (Speed Up Robust Features), that applies approximate Gaussian second derivative mask to an image at many scale for Hessian computation. Filters are shown in Figure 2.8b. The application of the filter is made faster by employing integral images, since only sums over rectangles are needed for the computation. The Hessian determinant is then employed for determining both keypoints location and scale. For rotation invariance, convolution with Haar wavelet filters (x and y directions) are computed for pixels sampled in a surrounding region. Both region size and sampling steps are proportional to the keypoint scale. Responses are then considered as (x, y) vectors, and an orientation window is slided computing the sum of all the vertical and horizontal responses of the vectors falling inside it, as shown in figure 2.8b. This gives a new two dimensional vector resuming the window content. The longest vector obtained by sliding the window represents the keypoint orientation.

Descriptors are computed by dividing the keypoints surrounding patches (oriented as keypoints directions) in regions, and computing wavelet responses. After weighting them with a Gaussian centered at the interesting point, wavelet responses are combined giving a 4-dimensional descriptor for each patch region, resulting in a 64-dimensional keypoint descriptor. The SURF algorithm has shown to be good in handling variation of scale and ro-

tation in different views, however it is less resistant than SIFT in situations in which illumination and 3D viewpoint change.

2.3.5 ORB

The ORB [10] (Oriented FAST and Rotated BRIEF) algorithm is an efficient alternative to SIFT and SURF, built on the FAST keypoins extractor and BRIEF descriptors. In particular it adds scale and orientation components to FAST corners, and rotation invariance to the BRIEF descriptors.

Scale invariance is reached by applying corner detection to the image pyramid levels. To return a number N of keypoints, the FAST threshold is set low enough to get a sufficient number of keypoints that are then ordered according to their Harris score. Then the first N of them are chosen. For orientation invariance the intensity centroid technique is used. The image centroid of a patch around a keypoint $\mathbf{k}$ is defined as:

$$\mathbf{c} = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right), \tag{2.28}$$

where the patch moments m are computed in the following way:

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y). \tag{2.29}$$

The assigned orientation is the angle of the $\mathbf{c} - \mathbf{k}$ vector. The descriptor orientation invariance is reached by applying BRIEF on the rotated test locations. To increase the discrimination capacity of the descriptors, the test locations are chosen using a statistical approach.

2.3.6 Features Matching

Features matching is the third step of the correspondences finding process, in which two sets of descriptors $\mathcal{D}_1$ and $\mathcal{D}_2$ are given, and the objective is to find for each element of $\mathcal{D}_1$ the most similar element in $\mathcal{D}_2$. Depending on the kind of descriptors, different similarity metrics are used. For SIFT and SURF descriptors the Euclidean distance is used, while for binary descriptors as BRIEF and ORB the Hamming distance is employed.

The most simple matching algorithm uses the brute-force approach, by comparing each $\mathcal{D}_1$ element with all the elements of $\mathcal{D}_2$. However this method is too expansive and can not be employed in real time applications.

For euclidean descriptors a speed up can be obtained by using K-d trees [11] (K-dimensional trees), that consist in organizing the elements of $\mathcal{D}_2$ (called training set) in a binary tree data structure in the following way.

Consider the i-th descriptors dimension, and the median m_i over all the data of such values. The set of descriptors is split in two and a tree node is built, assigning to the right sub-tree elements with i-th dimension greater than m_i, and to the left sub-tree the remaining ones. The process is iterated for each subset of points over all the remaining dimensions, resulting in a tree whose leaves corresponds to a bin of the descriptors space. When a query point in $\mathcal{D}_1$ is proposed, tree nodes are considered as decision nodes, bringing to the right sub-tree if the corresponding dimension element is greater than m_i and on the left sub-three otherwise. This process is iterated until a leaf is reached, and the nearest neighbor is searched among the $\mathcal{D}_2$ elements falling in the corresponding bin. For better accuracy, it is possible to backtrack some three levels and search for the nearest neighbor in more than one bin. This is done by considering an hyper-sphere around the the query point with radius equal to the best distance found up to now, and considering only the bins with some hyperplane side intersecting with the sphere. Advanced techniques for approximating and increasing performances of the matching are described in [12].

The K-d tree technique can not be used on binary features, because it is thought for euclidean spaces. A valid solution is LSH [13] (Local Sensitive Hashing), where points are stored in several hash tables and hashed in different buckets. Given a query descriptor, its matching buckets are retrieved and its elements compared using brute force matching.

Another approach, valid for both Euclidean and binary descriptors, is to build a tree by iteratively performing clustering over the training set descriptors, and when a query descriptor is proposed a leaf is reached by choosing at each level the node with centroid nearest to it. Then brute-force matching is applied to find the nearest training descriptor belonging to the reached leaf. Note that this approach can be used to trade-off accuracy and efficiency, by changing the number of children of each node and the number of tree levels.

2.3.7 Stereo Matching

The matching problem can be made faster when dealing with two images coming from a rectified stereo pair of cameras. In this situation the epipolar constraints simply says that two images of a scene point must be row aligned. Plus notice that the column pixel coordinate of the left camera perception must be bigger than the right one. An algorithm for matching keypoints in this settings is well explained in [14]. The algorithm first sorts both the left and right keypoints according to their row coordinates. Points on the same

rows are sorted according their column coordinates. For each left point the algorithm iterates over the right points until a feature on the same row is found. The nearest neighbor is searched among right features in the same row with lower column coordinate. When the row is exceeded, or a right point on the same row with higher column coordinate is found, a new left feature is considered. Thanks to sorting, the right point index never need to get back to previous rows when a new left point is asked for matching, making the matching algorithm very fast.

Less comparisons can be done if also a maximum disparity is allowed for matching features. The result is that all the features with a depth smaller than a threshold are discarded, thus the maximum disparity must be chosen according to the application and the stereo configuration.

2.4 Outliers Rejection

An outlier in a set of data is a sample that does not obey to the mathematical model best describing most of the elements of the set.

In many situations, we need to find the model parameters $\mathbf{w}$ that minimize a suitable error function $e(\cdot)$ over a set of sample $\mathcal{S} = \{s_i\}_{i=1..N}$ in presence of outliers:

$$\min_{\mathbf{w}} \sum_{\mathcal{S}} \mathcal{I}(s)\, e(\mathbf{w}, s), \tag{2.30}$$

where

$$\mathcal{I}(s) = \begin{cases} 1 & \text{if } s \text{ is an inlier,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.31}$$

Many computer vision problems require to compute the transformation relating two set of points (either image or world points) given their correspondences. However it is common that many of the computed correspondences are wrong, and need to be considered as outliers. Thus is essential to have robust estimation methods that detect and exclude them. In this section we describe one of the most successfully employed algorithm in the geometrical computer vision literature, called RANSAC. Then we show an example in which it is used to robustly locate a known planar object in an image.

2.4.1 RANSAC

RANSAC [15] (RANdom SAmple Consensus) is an iterative method that estimates the parameters of a mathematical model from a set of observed data contaminated by outliers. The algorithm steps are explained as follows.

Given a model that requires a minimum of n data points to compute its parameters, and a set of data points $\mathcal{S}$, randomly select a subset of n data points from $\mathcal{S}$ and compute an hypothetic model. Use the hypothetic model m to determine the subset $\mathcal{C}$ of points in $\mathcal{S}$ that are within some error tolerance t of m. The set $\mathcal{C}$ is called the consensus set of m. If $|\mathcal{C}|$ is greater than some threshold N, then $\mathcal{C}$ is a candidate for being the best consensus set. The process is iterated up to a finite number of iterations and the chosen inliers set is the consensus set with higher cardinality. If within the specified number of iteration no consensus set $\mathcal{C}$ such that $|\mathcal{C}| > N$ is found, the algorithm terminates with failure. After the best consensus set is found, it is possible to fit the model parameters over all its elements.

Other algorithms have been proposed following this approach, for instance MLESAC (Maximum Likelihood Estimation SAmple Consensus, [16]) that chooses as best consensus set the one with the higher likelihood, rather than the one with higher number of inliers. Another very simple variant of RANSAC is the following. Considering that RANSAC tries to maximize the cardinality of the consensus set, its objective be also thought as:

$$\min \sum_{s \in \mathcal{S}} \rho(e_s^2), \tag{2.32}$$

where $\rho(\cdot)$ is:

$$\rho(e^2) = \begin{cases} 0 & \text{if } e^2 < t^2, \\ \text{const.} & \text{otherwise,} \end{cases} \tag{2.33}$$

thus each outliers give constant penalty and inliers no score. By modifying the cost function as:

$$\min \sum_{s \in \mathcal{S}} \rho_2(e_s^2), \tag{2.34}$$

where $\rho_2(\cdot)$ is:

$$\rho_2(e^2) = \begin{cases} e^2 & \text{if } e^2 < t^2, \\ t^2 & \text{otherwise,} \end{cases} \tag{2.35}$$

a new sample consensus algorithm is obtained called M-SAC (M-estimator SAmple Consensus), that instead of maximizing the size of the consensus set chooses the consensus set that provides the minimum error, with no additional computational cost.

It easy to see that RANSAC is a probabilistic algorithm, thus it produces good results only with a certain probability. This depends on the number of iterations, the size of the minimal sampled set n, and the ratio

$$w = \frac{\#\text{ inliers}}{|\mathcal{S}|} \tag{2.36}$$

	percentage of outliers					
n	5%	10%	20%	30%	40%	50%
2	2	3	5	7	11	17
3	3	4	7	11	19	35
4	3	5	9	17	34	72
5	4	6	12	26	57	146

Table 2.1: Number of iterations needed for finding a solution with probability $p = 0.99$ of success, over different values n of sampling size and percentage of outliers.

representing the probability that a given sample is an inlier. The probability that n samples are inliers is w^n, and $1-w^n$ is the probability that at least one of them is an outlier. Performing k iterations, the probability p of success of the algorithm is obtained by setting $1 - p = (1 - w^n)^k$. From this the number of needed iterations for guaranteeing that the algorithm converges to the right solution with probability p is:

$$k = \frac{\log(1-p)}{\log(1-w^n)}. \tag{2.37}$$

We show in Table 2.1 the needed number of iterations for finding a good solution with probability of success $p = 0.99$, and different values of sample size and outliers percentage.

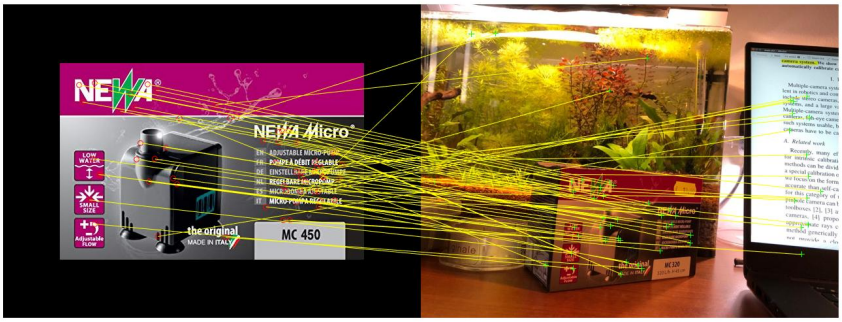
Equation 2.37 can be used to avoid performing all the prefixed RANSAC iterations if a confidence level is fixed. Indeed w can be approximated by substituting in Equation 2.36 the inliers number with the size of the biggest consensus set found up to now. Then if the fixed confidence is p, the algorithm stops as soon as $1 - (1 - w^n) \geq p$.

2.4.2 Example: Robust Homography Fitting

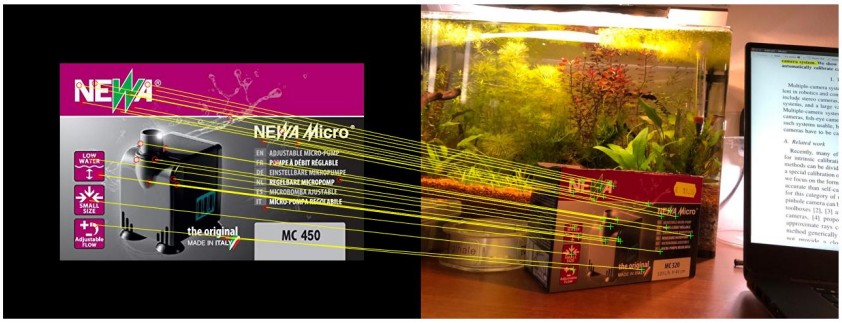
Here we show a simple application of the RANSAC algorithm together with keypoints extraction and matching for robustly find a known planar object in an image. Suppose the image P to be a representation of the planar object (sufficiently textured) we want to find in image I. We extract from both images features and compute their descriptors, then we match them. We obtain a set of keypoints correspondences $\{\langle \mathbf{x}_i^P, \mathbf{x}_i^I \rangle\}_{i=1..N}$ potentially comprising wrong matches. The elements of the inliers pairs are related by an homography $\mathbf{H}$ as:

$$\tilde{\mathbf{x}}_i^I = \mathbf{H}\,\tilde{\mathbf{x}}_i^P, \tag{2.38}$$

where points are expressed in homogeneous coordinates, and $\mathbf{H}$ is a 3×3 homogeneous matrix (the expressed relation does not change if we multiply



(a) All matches.



(b) Inliers.

Figure 2.9: Result of robust homography fitting between a model and a view of the same planar object. Model points are shown as red circles, and matched to view points represented by green crosses. Model and view matched points are joined by a yellow line. All obtained matches are shown in 2.9a, while the set of inlier matches is shown in 2.9b.

H by a constant factor). The matrix we want to find has nine elements but only eight DoF, thus it can be estimated from four correspondences in closed form. To find it in presence of wrong matches we run RANSAC, sampling four correspondences at a time. A result example of this procedure can be seen in Figure 2.9.

2.5 Stereo Camera Pose Estimation

Camera pose estimation is the problem of computing the six DoF camera extrinsic parameters in some reference system of interest.

Here we focus on the pose estimation of a calibrated stereo camera based on keypoints extraction and matching. The three main techniques employed

in the literature are called 2D-2D, 3D-2D, and 3D-3D localization. Their name depends on what kind of correspondences they use. For instance in the 2D-2D approach, keypoints correspondences are assumed to be available in two different views of the same scene and used for the computation of the quadrifocal tensor, that is the natural extension to four views of the fundamental matrix. The quadrifocal tensor can be then decomposed to compute the four camera projection matrices, containing information of relative cameras positioning. More details about the quadrifocal tensor can be found in [17].

In this thesis we focus on the other two mentioned approaches, that assume a set of 3D points of the observed scene to be given, and matched with their current perception. In the 3D-2D approach the 3D scene points are matched with their current images (for instance in the left camera), while in the 3D-3D approach with their reconstruction obtained by triangulation of their current perceptions.

2.5.1 Notation

We introduce the notation relative to quaternions computation, extensively used in the following section. A more detailed version of this section together with different rotation representations can be found in Appendix A.

A quaternion $\check{q}$ with real part q_0, and imaginary parts q_x, q_y, and q_z can be written as:

$$\check{q} = q_0 + iq_x + jq_y + kq_z. \tag{2.39}$$

The quaternions multiplication is obtained by the product of their expression, where the imaginary parts products follow the rules:

$$\begin{aligned} &i^2 = -1, && j^2 = -1, && k^2 = -1, \\ &ij = k, && jk = i, && ki = j, \\ &ji = -k, && kj = -i, && ik = -j. \end{aligned}$$

The product between two quaternions q and r can be also defined in matrix form:

$$\begin{aligned} \check{q}\check{r} &= \begin{bmatrix} q_0 & -q_x & -q_y & -q_z \\ q_x & q_0 & -q_z & q_y \\ q_y & q_z & q_0 & -q_x \\ q_z & -q_y & q_x & q_0 \end{bmatrix} = \mathcal{Q}\check{r}, \\ \check{r}\check{q} &= \begin{bmatrix} q_0 & -q_x & -q_y & -q_z \\ q_x & q_0 & q_z & -q_y \\ q_y & -q_z & q_0 & q_x \\ q_z & q_y & -q_x & q_0 \end{bmatrix} = \bar{\mathcal{Q}}\check{r}, \end{aligned} \tag{2.40}$$

where we called $\mathcal{Q}$ and $\bar{\mathcal{Q}}$ the matrices associated to respectively premultiplication and postmultiplication with quaternion $\check{q}$. Notice that the matrices associated with the conjugate quaternion are simply obtained by transposition. The conjugate of the quaternion $\check{q}$ is denoted as $\check{q}^*$, and is obtained by negation of the imaginary coefficients.

By thinking of quaternions as four dimensional vectors $(q_0, q_x, q_y, q_z)^T$ we obtain the quaternions dot product operation and quaternion norm. A quaternion is unitary if its norm equals 1.

Given a point $\mathbf{X} = (X\,Y\,Z)^T$, we can represent it as a purely imaginary quaternion $\check{X} = iX + jY + kZ$, and transform it using the unitary quaternion $\check{q}$ via composite product, obtaining a purely imaginary quaternion:

$$\check{X}' = \check{q}\check{X}\check{q}^*. \tag{2.41}$$

This operation preserves magnitude, dot product and cross product. As consequence the composite product by a unitary quaternion represents a rotation.

It is now useful to introduce the following identity:

$$\begin{aligned} \check{p}\check{q} \bullet \check{r} &= \bar{\mathcal{Q}}\check{p} \bullet \check{r} \\ &= \left(\bar{\mathcal{Q}}\check{p}\right)^T \check{r} \\ &= \check{p}^T \bar{\mathcal{Q}}^T \check{r} \\ &= \check{p}^T \check{r}\check{q}^* \\ &= \check{p} \bullet \check{r}\check{q}^*, \end{aligned} \tag{2.42}$$

that is used for derivation of the aligning transformation in the 3D3D approach.

2.5.2 The 3D-3D Approach

The 3D-3D method computes the extrinsic parameters by aligning a given set of 3D points representing the observed scene with the corresponding reconstruction, obtained by triangulation in the current perception. Thus we are given with a set of pairs $\{\langle \mathbf{X}_i^w, \mathbf{X}_i^c \rangle\}_{i=1..N}$ where $\mathbf{X}_i^w$ and $\mathbf{X}_i^c$ are the expression of the same scene point respectively in the world and camera reference systems.

The extrinsic parameters can be computed by solving the following problem:

$$\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{N} \left\| \mathbf{R}\mathbf{X}_i^w + \mathbf{t} - \mathbf{X}_i^c \right\|^2, \tag{2.43}$$

that takes the name of *absolute orientation*. Some of the most employed solution in the literature are [18–20]. Here we explain the solution presented in [18], that makes use of the quaternion representation for the rotation (see Appendix A). The algorithm also takes in account a scale factor between the two set of points, that we do not consider here because not useful in stereo settings.

Let us call $\mathbf{X}^w$ and $\mathbf{X}^c$ the centroids of world and camera sets of points. We define the new centered coordinates as $\mathbf{X}_i^{w'} = \mathbf{X}_i^w - \mathbf{X}^w$ and $\mathbf{X}_i^{c'} = \mathbf{X}_i^c - \mathbf{X}^c$. The best rotation is the solution of the following problem:

$$\max_{\mathbf{R}} \sum_{i=1}^{N} \mathbf{R}\mathbf{X}_i^{w'} \cdot \mathbf{X}_i^{c'}. \tag{2.44}$$

By representing the rotation as a unitary quaternion and points as purely imaginary quaternions, the problem becomes:

$$\max_{\check{q}} \sum_{i=1}^{N} \check{q}\check{X}_i^w \check{q}^* \cdot \check{X}_i^c. \tag{2.45}$$

By using Equation 2.42 we obtain:

$$\max_{\check{q}} \sum_{i=1}^{N} \check{q}\check{X}_i^w \cdot \check{X}_i^c \check{q}. \tag{2.46}$$

Following the notation introduced in Section 2.5.1, we call respectively $\mathcal{X}$ and $\bar{\mathcal{X}}$ the matrices representing the pre-multiplication and post-multiplication operators associated to the quaternion $\check{X}$. We obtain:

$$\begin{aligned}
\max_{\check{q}} \sum_{i=1}^{N} \check{q}\check{X}_i^w \cdot \check{X}_i^c \check{q} &= \max_{\check{q}} \sum_{i=1}^{N} \bar{\mathcal{X}}_i^w \check{q} \cdot \mathcal{X}_i^c \check{q} \\
&= \max_{\check{q}} \sum_{i=1}^{N} \left(\bar{\mathcal{X}}_i^w \check{q}\right)^T \mathcal{X}_i^c \check{q} \\
&= \max_{\check{q}} \sum_{i=1}^{N} \check{q}^T \bar{\mathcal{X}}_i^{wT} \mathcal{X}_i^c \check{q} \\
&= \max_{\check{q}} \check{q}^T \left(\sum_{i=1}^{N} \bar{\mathcal{X}}_i^{wT} \mathcal{X}_i^c \right) \check{q} \\
&= \max_{\check{q}} \check{q}^T \mathbf{N} \check{q}.
\end{aligned} \tag{2.47}$$

The author shows that the unit quaternion maximizing the quadratic form $\check{q}^T\mathbf{N}\check{q}$ equals the eigenvector corresponding to the most positive eigenvalue of the matrix $\mathbf{N}$. Interestingly, the elements of the matrix $\mathbf{N}$ can be

computed as sum and differences of the elements of another matrix, that we call $\mathbf{M}$:

$$\mathbf{M} = \sum_{i=1}^{N} \mathbf{X}_i^{w'} \mathbf{X}_i^{c'T}. \tag{2.48}$$

After the rotation matrix $\mathbf{R}$ is obtained from $\dot{q}$, the translation vector is easily computed as:

$$\mathbf{t} = \mathbf{X}^c - \mathbf{R}\mathbf{X}^w. \tag{2.49}$$

Note that this algorithm gives both a minimal solution ($N = 3$) to be employed inside an outliers rejection scheme, and a solution to the overconstrained problem ($N > 3$). Suppose for instance that the set $\{\langle \mathbf{X}_i^w, \mathbf{X}_i^c \rangle\}_{i=1..N}$ contains 30% of outliers, the number of iterations needed to find a good inlier set with probability $p = 0.99$ is:

$$\frac{\log(1 - 0.99)}{\log(1 - 0.7^3)} = 10.9628,$$

thus the number of iterations to perform is 11.

2.5.3 The 3D-2D Approach

Suppose that a set of pairs $\{\langle \mathbf{X}_i^w, \mathbf{x}_i \rangle\}_{i=1..N}$ is given, where $\mathbf{X}_i^w$ is a 3D scene point expressed in the world reference system and $\mathbf{x}_i$ its current camera perception from a single calibrated camera. The problem of finding the extrinsic parameters $\mathbf{R}$, $\mathbf{t}$ such that

$$\mathbf{x}_i = \mathbf{K}\,[\mathbf{R} \mid \mathbf{t}]\,\mathbf{X}_i^w \quad \forall\, i = \{1..N\} \tag{2.50}$$

is known as *perspective-n-points* (PnP). In general the equations can not be simultaneously satisfied due to noise, thus the problem is formulated in least square fashion.

To deal with outliers (wrong correspondences) a minimal solution is employed inside an outliers rejection scheme. The minimum number of required correspondences is three, since the extrinsic parameters have six DoF and each pair provides two equations:

$$\begin{aligned} u &= \frac{r_{11}X + r_{12}Y + r_{13}Z + t_1}{r_{31}X + r_{32}Y + r_{33}Z + t_3}, \\ v &= \frac{r_{21}X + r_{22}Y + r_{23}Z + t_2}{r_{31}X + r_{32}Y + r_{33}Z + t_3}, \end{aligned} \tag{2.51}$$

where (u, v) are normalized image coordinates. A fast and stable solution to the P3P problem has been proposed in [21]. The algorithm involves a large number of elimination steps, which we do not discuss in this thesis.

Four possible solutions are returned thus a fourth correspondence is needed to disambiguate and choose the one with smaller reprojection error.

The general case (n points) has no closed form solution, and many iterative and non-iterative algorithm have been proposed in the literature. Some examples of iterative solutions are [22, 23], while non-iterative solutions has been proposed in [24–26].

Here we introduce the orthogonal iteration algorithm [22] (generally called LHM, from the authors names), that starts from a guess and iteratively refines it until convergence. Let us assume that the $\mathbf{x}_i$ image points are expressed in normalized image coordinates. These can be thought as vectors in the camera frame, each one lying on the corresponding scene point viewing ray. For each correspondence the LHM method makes use of the following object space error function:

$$e_i = (\mathbf{I} - \mathbf{F}_i)(\mathbf{R}\mathbf{X}_i^w + \mathbf{t}), \tag{2.52}$$

where $\mathbf{I}$ is the 3×3 identity matrix, and $\mathbf{F}_i$ is the projection matrix on vector $\mathbf{x}_i$:

$$\mathbf{F}_i = \frac{\mathbf{x}_i \mathbf{x}_i^T}{\mathbf{x}_i^T \mathbf{x}_i}. \tag{2.53}$$

The intuition behind Equation 2.52 is that scene points expressed in the camera reference frame lie on the corresponding viewing ray, thus they are unaffected when projected on it. The authors show that the minimum of the error function

$$E(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^{N} \|e_i\|^2 \tag{2.54}$$

can be iteratively obtained starting from a rotation guess. Suppose that the current rotation solution is $\mathbf{R}^{(k)}$, the corresponding translation $\mathbf{t}^{(k)}$ has closed form solution, since 2.52 is linear in $\mathbf{t}$. Then the new rotation $\mathbf{R}^{(k+1)}$ is computed by solving the following absolute orientation problem:

$$\mathbf{R}^{(k+1)} = \arg\min_{\mathbf{R}} \sum_{i=1}^{N} \left\| \mathbf{R}\mathbf{X}_i + \mathbf{t}^{(k)} - \mathbf{F}_i \left(\mathbf{R}^{(k)}\mathbf{X}_i + \mathbf{t}^{(k)} \right) \right\|^2, \tag{2.55}$$

and the process is iterated up to convergence.

Another widely used solution for reprojection error refinement starting from an extrinsic parameters guess is to use iterative gradient based optimization methods such as Gauss-Newton (GN) or Levenberg-Marquardt (LM). Since these methods performs unconstrained optimization, the matrix representation of the rotation is not well suited. A better approach is to perform optimization on the special Euclidean group (more details about on

manifold optimization can be found in [27]). However, in most applications, the minimal Rodrigues representation (se Appendix A) works well if a good starting guess is provided.

2.6 VSLAM and Visual Odometry

The camera pose estimation problem finds its main applications in robot localization and 3D scene reconstruction. The use of keypoints for localization is a valuable choice in both problems, because they allow to translate the redundant dense information contained in an image to a sparse and stable representation, speeding up the localization process.

In the robot localization scenario, a camera is mounted on the robot, and its pose is recovered from the knowledge of the rigid transformation relating the camera and robot frames. Two different robot localization problems can be solved by employing a camera: odometry and SLAM (simultaneous localization and mapping).

The odometry problem consists in computing the robot pose incrementally, composing transformations relating positions at discrete consecutive timestamps. When a camera is employed this takes the name of *visual odometry* (VO), and the camera pose estimation problem is solved between consecutive image frames. When the 3D3D approach is used, sparse scene reconstruction at time t and $t+1$ are obtained by keypoints extraction, followed by stereo matching and triangulation. Then features coming from two different timestamps are matched, and point cloud alignment is performed, giving the camera motion.

In general, wrong correspondences are rejected using RANSAC, with a minimum of 3 points sampling as explained in Subsection 2.5.2. The 3D2D approach works in a very similar way, however, instead of aligning sparse scene reconstruction, the perspective-n-points problem is solved among the 3D points obtained at time t and the image features obtained at time $t+1$, using P3P as minimal solution for outliers rejection. These two methods are the mostly used because performing 3D reconstruction allows to speed up consistently the matching problem. Suppose that a guess of the extrinsic parameters $\mathbf{R}_g$, $\mathbf{t}_g$ is given. We call with a little abuse of notation t the reference system representing the camera position at time t. For each 3D point $\mathbf{X}_i^t$ we obtain its projection on the next image as:

$$\mathbf{x}_{g,i} = \mathbf{K}\left[\mathbf{R}_g \mid \mathbf{t}_g\right]\mathbf{X}_i^t, \tag{2.56}$$

then, each 3D point $\mathbf{X}_i^t$ is matched by choosing the current image point with most similar descriptor inside a small region around the guessed projection

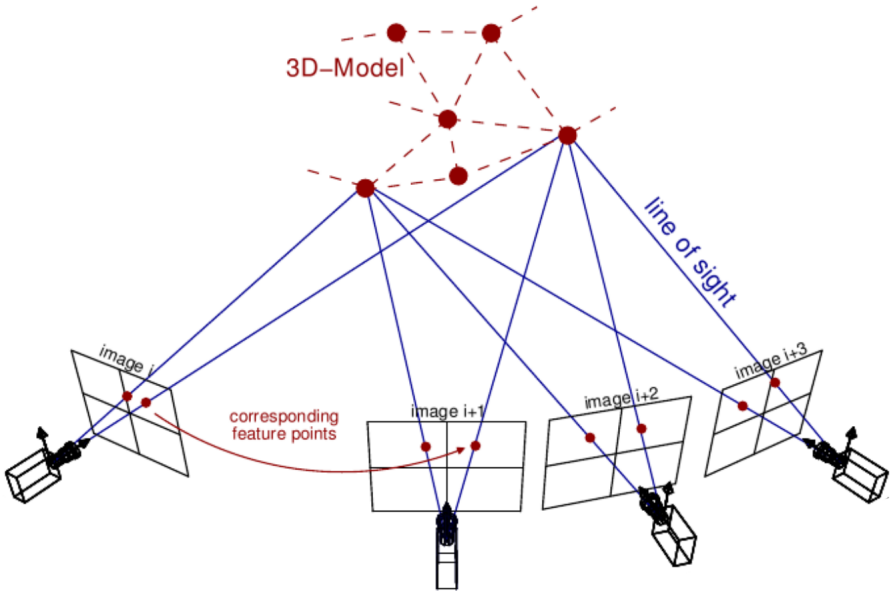


Figure 2.10: A representation of the bundle adjustment problem, where a camera observes the same scene in four different poses over time. By features matching 3D points are associated with all their perceptions. 3D points and camera poses are then jointly refined in non linear reprojection error minimization fashion.

$\mathbf{x}_{g,i}$. A minimum similarity threshold is employed for rejecting bad matches. For obtaining the extrinsics guess, is in general sufficient to employ the constant motion model if the frame rate is sufficiently high, considering the extrinsic parameters computed in the step from $t-1$ and t.

The 3D2D approach is generally preferred with respect to point clouds alignment in VO applications, and the motivation is based on the shape of the uncertainty region around triangulated points, that we intuitively explained in Section 2.2.4. Indeed the 3D3D method suffers of depth error arising in triangulation, specially when a small baseline stereo camera is used. By projecting 3D points on the image plane, the 3D2D method reduces this effect resulting in higher pose estimation accuracy when the point of view smoothly changes between subsequent perceptions.

A widely employed technique for improving the estimated trajectory accuracy is *bundle adjustment* (BA), consisting in joint refinement of camera poses and 3D points. The problem has expression:

$$\min_{\mathbf{R}_t^w, \mathbf{t}_t^w, \mathbf{X}_i^w} \sum_i^N \sum_t^T \left\| \mathbf{x}_{i,t} - \pi\left(\mathbf{K}, \mathbf{R}_w^t, \mathbf{t}_w^t, \mathbf{X}_i^w\right) \right\|^2, \tag{2.57}$$

where $\{\mathbf{X}_i^w\}_{i=1..N}$ is the set of observed scene points, expressed in a global

world frame (generally the initial camera pose), $t = 1..T$ is the timestamp index, $\mathbf{R}_w^t$, $\mathbf{t}_w^t$ are the extrinsic parameters at time t, and $\mathbf{x}_{i,t}$ are the pixel coordinates of the perception of point $\mathbf{X}_i$ at time t. The function $\pi(\cdot)$ is the projection function, computing image pixel coordinates of the 3D point projection. This problem has in general many variables to be optimized, however presents a very sparse Jacobian. Thus a different version of LM that exploits Jacobian sparsity is employed, called sparse LM. For real time applications only a window of previous n perceptions is considered in the optimization problem, and the algorithm takes the name of *windowed bundle adjustment* (WBA). A representation of the BA problem can be seen in Figure 2.10, where the same scene model is observed in different camera poses, and features correspondences among multiple views are known.

It is well known that odometry presents in general drift in the estimated trajectory with respect to the true one, as the traversed path length increases. Indeed odometry is only concerned with the local consistency of the estimated trajectory. SLAM can be considered as an extension of odometry, that computes the robot trajectory together with a map of the observed environment. SLAM algorithms are characterized by a process called loop closing, used to guarantee global consistency of the estimated map and trajectory. This consists in detecting when the robot is in an already visited place and localizing it in the map constructed up to now, increasing the information exploited in the map optimization procedure. In general SLAM systems are based on probabilistic approaches and consider also the confidence of the map and trajectory estimation. When the SLAM process is finished the map is stored and can be used for robot localization, and eventually trajectory planning.

Visual SLAM (V-SLAM) is the application of camera sensors to the SLAM problem. When keypoints are used the map is composed by the set of 3D triangulated features, associated with their visual descriptors. Incremental motion is estimated in VO fashion, and loop closure is performed by visual place recognition followed by a geometrical localization step. Here we do not go into the details of place recognition, we only say that after this procedure a set of 3D map points is given, and the geometrical step consists in localize the camera with respect to them either with a 3D3D or 3D2D approach. This serves both as a check on the correctness of the place recognition output (the localization procedure has finished with a good number of inlier correspondences), and as an additional constraint on the map and the trajectory. Note that the map obtained in this way is only a sparse representation of the environment, not particularly useful for planning. However sparse 3D reconstructions associated with visual descriptors

are very effective for localization purposes. A dense reconstruction of the map can be obtained by performing dense stereo matching and triangulation, that however is not concern of this thesis.

Another application, that takes the name of *structure form motion* (SfM) in the computer vision literature, consists in recovering the 3D structure of the observed scene together with the camera poses in the different observations, from a set of ordered or unordered scene perceptions. SfM is more general than VO, that it is only concerned with the robot trajectory, however the approaches to VO we just introduced can be thought as an application of SfM, with the only difference that images are required to be in sequence at an high frame rate (for keypoint tracking), and the procedure is performed in real time.

Chapter 3

State of the art

The aim of this chapter is to present solutions from the literature to the camera pose estimation problem. Since the objective of this thesis is to develop localization algorithms for a system of multiple rigidly attached stereo cameras, we initially focus on stereo camera localization in Section 3.1. Then we explore some approaches to the localization of multi-camera systems in Section 3.2. Finally, in Section 3.3, we resume different solutions to the multi-camera calibration problem, that consists in determining the spatial transformation relating the frames of the cameras belonging to the rig.

3.1 Stereo Camera Localization

In this section we give an overview of two different keypoint feature based VSLAM algorithms: ProSLAM [14] and ORB-SLAM 2 [28]. The choice of introducing two VSLAM systems is motivated by the fact that they need to solve the relative camera pose estimation problem in both incremental motion estimation, where the view smoothly changes, and in the loop closing situation, in which there might be a wide change in the point of view. ProSLAM (Programmer SLAM) is a lightweight stereo visual SLAM system, built to provide an easy to understand pipeline divided in four modules as shown in Figure 3.1: triangulation, incremental motion estimation, map management and relocalization. In the triangulation module pair of rectified undistorted stereo images are processed to perform a sparse reconstruction of the observed scene, by triangulating FAST features associated to BRIEF descriptors. An evenly distributed set of left image features is selected, and then matched in the right images by epipolar search. The point reconstruc-

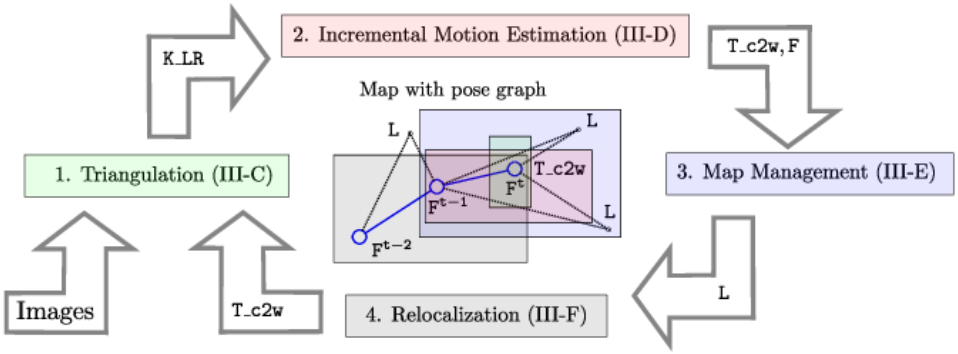


Figure 3.1: The ProSLAM pipeline [14].

tion is obtained by triangle similarity as explained in Section 2.2.4.

The incremental motion estimation module processes the reconstruction obtained at frame $t-1$ together with image points survived to stereo matching at frame t. 3D2D matchings are obtained by reprojection on left image under constant motion model as explained in Section 2.6, and an iterative on-manifold optimization procedure minimizes the reprojection error while rejecting correspondences presenting error higher than a fixed threshold at each step. During the optimization procedure, the translational contribution of points that are considered too far from the camera is disabled. This is because for such points the depth can not be reliably estimated due to the small disparity.

Inliers image points and the estimated relative position are then passed to the map management module, that adds them to the current local map. A local map is a data structure composed of a set of triangulated points in subsequent frames, associated with their perceptions and the corresponding camera positions. The map management module recovers inliers that the previous module has lost, plus refines the 3D position of tracked points by running an information filter for each of them. Additionally, this module is given with the task of creating a new local map when a certain amount of translation or rotation has been done with respect to the current local map position.

The relocalization module finally solves the loop closure problem, by performing visual place recognition among the current and the previous local maps, followed by a geometrical localization check. Place recognition is accomplished using HBST [29] (Hamming Distance embedding Binary Search Tree). This algorithm performs both fast binary features matching and similar image retrieval from a set of descriptors, however in this case the set of descriptors belonging to each local map are used. The geometrical

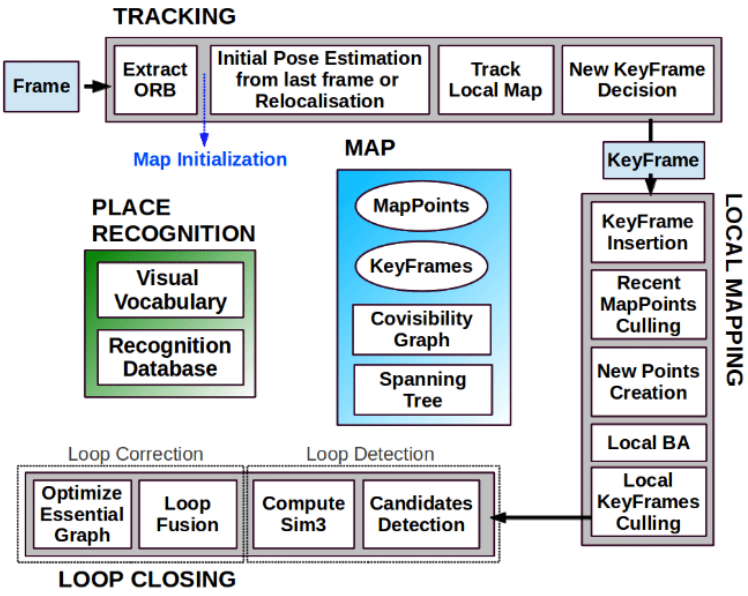


Figure 3.2: The ORB SLAM 2 pipeline [28].

localization step consists in point clouds alignment with ICP (Iterative Closest Point), and is applied between the current map and all the stored local maps with a sufficiently high similarity score. For the map optimization no bundle adjustment procedure is applied, instead local maps are arranged in a pose graph that is optimized using g2o [30].

ORB SLAM 2 is a visual SLAM algorithm built for monocular, stereo and RGB-D cameras, based on the previously released monocular version ORB SLAM [31]. As the name suggests, ORB features are extracted, however the image is discretized in cells and the same number of keypoints is retained in each cell. This is done to obtain a set of evenly distributed features in the image, that has shown to improve odometry performance in the literature. In the stereo settings left and right features are matched by epipolar search, assuming rectified images. The algorithm distinguish among three kinds of features: close stereo, far stereo, and monocular. Close stereo points have a depth that is smaller than 40 times the baseline, and can be safely triangulated from a single frame. Far stereo points are instead characterized by higher depth, and are triangulated only when supported by multiple views. Finally, monocular keypoints are the ones for which no stereo matching has been found, and are triangulated when tracked in multiple views. Different modules of the ORB SLAM 2 algorithm are shown in Figure 3.2. The tracking procedure estimates the motion among subsequent

frames.

For each ORB feature with an associated 3D point, a match in the current image is searched in a small patch around it. Outliers are discarded by and orientation consistency test among ORB features in previous and current image, and the new position is solved in 3D2D fashion using ePnP [24]. This process gives an estimation of the current pose in the map, and allows to project the set of points belonging to the current local map on the current image, increasing the number of matches. At this point a motion only BA procedure (map points are not optimized) is applied inside the current local map, to refine the pose estimation by using both monocular and stereo keypoints. When a local map is completed, BA is applied to refine 3D points.

Loop closing is performed using the bag of world approach of DBoW2 [32], that also performs fast descriptors matching. After a loop closure candidate is detected, 3D3D matchings are available and point clouds alignment is performed inside the RANSAC algorithm with the closed form solution proposed in [18]. In a separate thread a full BA procedure optimize the map, and is stopped when a loop is detected. After a loop is successfully closed the optimization procedure restarts.

3.2 Multi-Camera Localization

In this section we resume some state of the art solutions to the multi-camera pose estimation problem, in particular when the cluster is composed of rigidly attached cameras with non-overlapping FOVs. Two main approaches arises in the literature: the decoupled one, in which cameras poses are solved independently and then joined, and the coupled one, where the whole pose rig is solved using simultaneously the information captured by each camera. A decoupled VO solution has been proposed in [33], where the trajectories of two cameras facing opposite directions are independently computed and merged. The fusion step is accomplished expressing both estimates in a common reference frame, and computing their covariances. The covariance of each trajectory estimate has shown to be the inverse of the Hessian in the reprojection error function optimum, up to a scale factor. The absolute pose however can be obtained by imposing the rigid body constraint over the two camera trajectories. A filtering procedure takes into account covariances for estimating the whole rig motion from the two, and each camera motion is then recovered by assuring the satisfaction of the rigid body constraint. The full pipeline is resumed in Figure 3.3.

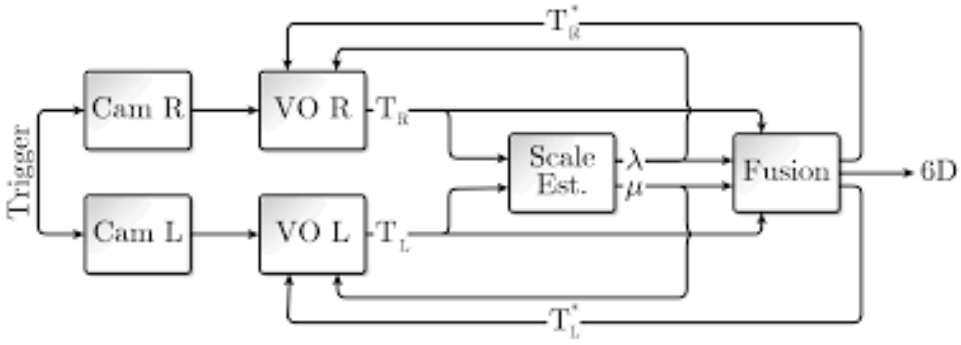


Figure 3.3: Pipeline of the decoupled VO solution proposed in [33].

Another decoupled solution has been proposed in [34], where two back-to-back stereo cameras are used, and each camera has two extended Kalman filter, one for estimating the camera pose and one for estimating the map.

The last decoupled solution we mention is the one developed in [35, 36], in which two stereo cameras, together with an IMU are fused for visual odometry. FAST corners are associated with BRIEF descriptors and two stereo cameras compute odometry independently. 3D2D matches are obtained between subsequent frames by projecting 3D points on the current image, where cameras projection matrices are derived from the IMU sensed motion. The PnP problem is then solved by assuming the IMU rotation to be correct, leading to the computation of only the translation factor as a linear least square problem. For outliers rejection the authors propose a new scheme called LONSC (LONgest Successive Consistency), that places samples in a one dimensional array and considers as inlier set the longest sequence of successive elements voting for the same motion. After the longest set is found another iteration over the array is performed for adding to the inliers set correspondences that have not been found in the first iteration. Different odometers are then fused by complementary filtering, considering the inliers number and their depth in each motion estimation for weighting.

A multi-camera VSLAM algorithm called MCPTAM (Multi-Camera Parallel Tracking And Mapping) has been proposed in [37–39]. This is based on the previous monocular version PTAM [40], that performs keypoints tracking along subsequent frames in the tracker module, and builds the environment map optimized with full BA in a separate module called mapper. The tracker computes camera incremental motion by matching visible map points with their current images. Correspondences are obtained by reprojection under simple motion model, and the pose is refined by non linear optimization. The mapper increases the number of correspondences by pro-

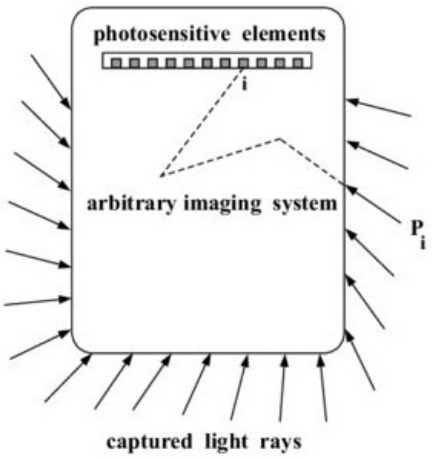


Figure 3.4: A representation of a generalized imaging model, in which light rays take arbitrary paths before meeting photosensitive elements which form the image [43].

jecting map points on the current image, and triangulate points along multiple views. Bundle adjustment is performed locally on the map for further refinement. Note that using only one camera PTAM needs an initialization phase in which the first movements scale are computed.

MCPTAM extends by employing a system of ultra wide FOV cameras, requiring to change the camera model and reprojection error Jacobian. The steps performed by the monocular tracking unit are applied on each camera independently for finding correspondences, however the pose refinement is performed in coupled way by reprojection error minimization. By projection of map points on the current view the situation in which multiple cameras observe the same point along the track is captured, and the bundle adjustment procedure is adapted to the multi camera settings. Other examples of pose estimation by joint reprojection error minimization on all cameras can be found in [41] and [42].

Higher coupling has been reached with the generalized imaging model (GIM) proposed in [43]. This model is able to represent any sensing system that performs a mapping from incoming scene rays to photosensitive elements, by abstracting away from which path light takes and treating the sensor as a black box. A representation of a general imaging model can be seen in Figure 3.4, in which incoming light rays take arbitrary paths before being sampled. Multi-camera sensors fall inside this class since, being a system of more than one camera, they perform a scene projection on more than one center. The use of multi-camera sensors in this framework is analyzed in [44], allowing to use a network of cameras as if they were a single

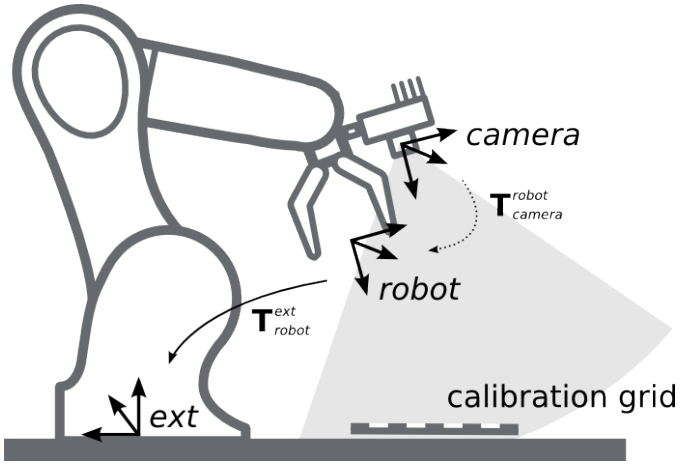


Figure 3.5: Hand eye calibration for the computation of the transformation relating the robot gripper to the camera. The gripper and camera trajectories are computed independently and used to form rigid body constraints.

imaging device. Solutions to the generalized perspective-3-points problem (gP3P) has been developed in [45, 46], and the n-points problem (gPnP) case is solved in [47–49]. In [50] both the minimal and non-minimal problem are solved, and on this the monocular SFM pipeline is applied to a pair of back-to-back facing cameras. A bootstrapping step solves for the scale as in [33], then when the scale is estimated the camera pair is considered as a single generalized imaging device. Points are triangulated among multiple views and expressed in a single reference system, and motion is recovered in 3D2D fashion by running gP3P inside RANSAC followed by gPnP over the inliers. Then WBA is applied with a window size of 2, including the scale factor in the optimized function.

3.3 Multi-Camera Calibration

Both coupled and decoupled solutions require the transformation relating different cameras of the cluster to be known. In the decoupled approach calibration is needed in order to express different camera poses in a common reference frame, while in coupled approaches all the views are resumed in a single world reconstruction, that is then reprojected on each camera. Here we introduce calibration approaches applied to camera systems with non overlapping FOVs. When instead different visual fields overlap enough, classical stereo calibration procedures can be employed. Two main classes of self-calibration procedures arises in the literature: pose-only approaches

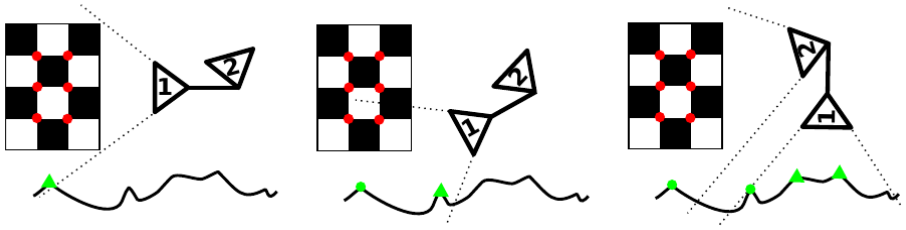


Figure 3.6: The multi-camera calibration procedure of MCPTAM, in which the first camera observes a checkerboard pattern for initialization of monocular SLAM, then the second camera il localized and moved in the built map for obtaining scene points association over different views [38].

and mapping-based approaches. The first one consists in performing either VO or VSLAM independently on each camera, and then applying hand-eye calibration to the estimated trajectories. Hand-eye calibration takes its name from the robotics community, as the computation of the Euclidean transformation relating a gripper (hand) frame to a camera (eye) frame. The situation in which the camera is attached to the gripper is shown in Figure 5.8. Camera and gripper trajectories are simultaneously computed and the rigid body constraint forms a set of equations that is then solved for the relating transformation; however this straightforwardly applies to a pair of cameras.

This technique has been employed in [51] where different cameras are arranged in such a way they share the optical center, considering the whole rig as a spherical camera. Due to the strong positioning assumption, only rotations need to be estimated thus a monocular SFM procedure with no absolute scale recovering is employed for motion computation. In [52] another procedure based on SFM is proposed. From trajectories defined up to a scale the cameras relative orientation is solved, leaving the absolute pose estimation as a linear least square problem. Finally, a non-linear optimization procedure refines relative extrinsic parameters.

Mapping-based approaches consist instead in performing environment mapping together with trajectory computation independently for each camera, followed by joint reprojection error minimization in BA fashion. During the optimization procedure each camera pose is expressed with respect to a rig frame (generally one of the cameras frames), thus the absolute pose of each camera for reprojection is expressed as composition of rig pose and relative camera pose. In [53] sparse maps of SURF features are independently generated with monocular VSLAM, and then aligned with 3D similarities

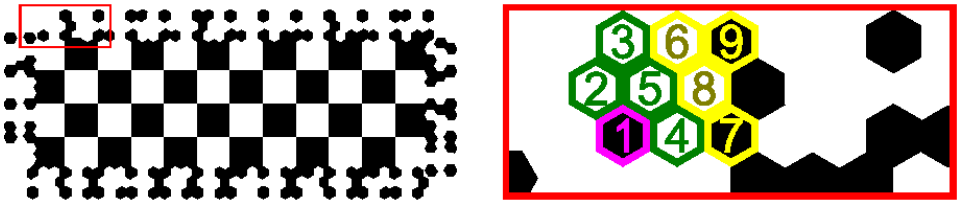


Figure 3.7: An example of calibration pattern used in [55]. Many of this patterns are used to calibrate a multi-camera with non overlapping FOVs. Binary markers around the checkerboard are used for solving data association and making the checkerboard pattern unambiguous.

found with absolute orientation inside RANSAC. At this point a single map defined up to a scale is available, where each 3D point is associated with perceptions potentially coming from more than one camera, and full BA is applied for refining map points, rig poses, and relative camera poses up to a scale.

In [38] an initialization step is included, in which the first camera observes a known planar pattern. Keeping the pattern fixed, the rig is moved and a monocular VSLAM algorithm builds the environment map with known scale. At this point the second camera is moved in order to observe the planar pattern, so that its pose in the already built map is estimated. By moving the rig in the environment, map points built by the first camera are found in the second camera obtaining 3D2D matches. The procedure is iterated until the environment map is built and enough correspondences are found for each camera. From trajectories a calibration guess is obtained, and then refined with full BA. A scheme of the procedure can be seen in Figure 3.6

A third class of approaches is the photogrammetric one, in which one or more known planar patterns are employed. A pair of planar pattern is placed on a rod in [54], in such a way their distance can be adjusted depending on the geometry of the cluster to be calibrated. Using single camera photogrammetric techniques the relative position of the two markers is estimated, and finally the pattern is used for the calibration of two cameras in a similar manner to stereo camera calibration.

In [55] a set of checkerboard patterns are arranged, each one surrounded by a binary marker. Thanks to this choice, high accuracy corners detection can be employed, and global data association is easily solved. The rig trajectory, pattern points, and relative camera poses are then estimated with BA. An example of checkerboard surrounded by binary markers is shown in Figure 3.7.

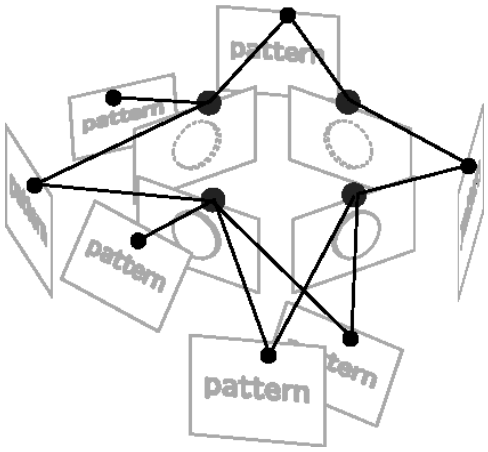


Figure 3.8: An example of pose graph obtained in [56] for the calibration of a multi-camera sensor composed of four cameras with non overlapping FOVs. A pattern node represents a pose of the pattern at some time, and is joined to a camera node if it falls in the FOV of the corresponding camera.

In [56] a single planar pattern is used, under the assumption that two cameras of the rig can simultaneously observe some part of it, even if their FOVs do not overlap. This requires the pattern to be unambiguous and easy to detect even if not fully inside the camera FOV. To this extent the authors develop a custom pattern by reverse engineering of SURF extractor, in such a way it provides an high number of distinctive features. In this way many common views of a known scene are obtained, and BA is employed for refinement after arranging pattern and cameras poses in a pose graph as shown in Figure 3.8.

Chapter 4

Multi-Stereo Pose Estimation

In this chapter we present the algorithms developed in this thesis and the settings in which they can be used. In Section 4.1 we explain the assumptions made on the kind of used sensor, plus the nomenclature we use to refer to single camera elements and reference systems placement. In Section 4.2 we explain the scene reconstruction procedure from a single perception. In Section 4.3 we introduce the matching procedures used for obtaining correspondences between a scene model and its perception, and in Section 4.4 we explain relative pose estimation algorithms based on 3D3D and 3D2D approaches. As a proof of concept for the applicability of reconstruction, matching, and relative pose estimation procedures, we introduce a simple VO pipeline based on them in Section 4.5. In Section 4.6 we explain our multi-stereo calibration procedure, fundamental step since all pose estimation algorithms make the assumption of calibrated multi-stereo camera. Finally in Section 4.7 we give some details about our implementation of the algorithms.

4.1 Notation

We develop calibration and relative pose estimation algorithms for a system of multiple stereo camera sensors with non overlapping FOVs. This means that the pair of cameras belonging to the same stereo sensor observe the same scene, however different stereo cameras can not observe simultaneously the same world region. Plus we assume the cameras to be synchronized, meaning that they capture images all at the same time.

To each stereo camera we assign a numerical ID $c \in \{1..C\}$ and a reference system placed on the corresponding left camera. When needed, we refer to single cameras in a stereo pair by their positioning (left and right). We place the rig reference system on the first stereo camera, called principal stereo, and we call principal camera the left element of such pair. The rig pose in a generic reference frame f is represented as $\mathbf{R}^f_{rig}$, $\mathbf{t}^f_{rig}$. The calibration matrices of the c left and right cameras are referred as $\mathbf{K}_{cl}$ and $\mathbf{K}_{cr}$, while their positioning in f as $\mathbf{R}^f_{cl}$, $\mathbf{t}^f_{cr}$ and $\mathbf{R}^f_{cr}$, $\mathbf{t}^f_{cr}$. We consider the sensor to be calibrated when all the intrinsics are known, plus the position of each camera $\mathbf{R}^{rig}_c$, $\mathbf{t}^{rig}_c$ in the rig is known. For a calibrated multi-stereo camera we define the projection functions:

$$\begin{aligned} \pi_{cl}(\mathbf{R}, \mathbf{t}, \mathbf{X}) &= \pi(\mathbf{K}_{cl}, \mathbf{R}^{cl}_{rig}\mathbf{R}, \mathbf{R}^{cl}_{rig}\mathbf{t} + \mathbf{t}^{cl}_{rig}, \mathbf{X}), \\ \pi_{cr}(\mathbf{R}, \mathbf{t}, \mathbf{X}) &= \pi(\mathbf{K}_{cr}, \mathbf{R}^{cr}_{rig}\mathbf{R}, \mathbf{R}^{cr}_{rig}\mathbf{t} + \mathbf{t}^{cr}_{rig}, \mathbf{X}), \end{aligned} \tag{4.1}$$

as the projection of point $\mathbf{X}$ on left and right cameras of stereo pair c when the principal camera extrinsics are $\mathbf{R}$, $\mathbf{t}$.

An acquisition of a multi-stereo sensor is the set of images simultaneously obtained by all camera at some time instant. We refer to the images acquired by the c stereo camera as the pair $\langle I^{cl}, I^{cr} \rangle$.

Since our algorithms only make use of interesting points found in the images, the latter are discarded after extraction and we refer to a perception as the set of all found keypoints together with their descriptors, each one associated with the ID of the perceiving camera. We refer to a keypoint found in I^{cl} as $\mathbf{x}^{cl}$ and in I^{cr} as $\mathbf{x}^{cr}$.

4.2 Reconstruction

To develop relative pose estimation algorithms for a multi-camera sensor that can be employed in a classical stereo localization pipeline, we need to define a reconstruction procedure computing observed scene points in the sensor frame from a single view. Given the multi-stereo configuration, this is simply the union of all the stereo reconstructions expressed in the rig frame, that can be obtained under the assumption of calibrated sensor.

Assuming each stereo pair to provide rectified images, we apply the same epipolar search procedure of [14] that we report in Algorithm 1. However to handle imprecisions in the rectification process and sub-pixel precision of extracted points, we substitute all the numerical comparisons relating keypoints row coordinates with soft comparisons. In particular we define a constant parameter that represents the epipolar lines thickness, and we

consider two points to be on the same horizontal line if their image row coordinates distance does not exceed half of the thickness. Formally:

$$\begin{cases} a \prec b & \text{if } a < b - \frac{\text{epipolar thickness}}{2}, \\ a \approx b & \text{if } |a - b| \leq \frac{\text{epipolar thickness}}{2}, \\ a \succ b & \text{otherwise.} \end{cases}$$

The algorithm might miss some matches due to soft comparisons, however it has shown to work well for small values of epipolar thickness; also bad correspondences are often present. A possible way of reduce bad correspondences is to triangulate and then reproject back points on one of the images, to remove stereo pairs giving a reprojection error higher than a threshold (either fixed or adaptive, e.g. mean of errors). Anyway subsequent steps of the localization pipeline perform robust outliers rejection, thus this step can be skip. We report in Figure 4.1 a visual comparison of the epipolar search and brute force algorithms for two different stereo acquisitions.

When stereo matches are obtained, we perform triangulation. To directly obtain 3D points in the rig frame, we apply the linear triangulation method introduced in Subsection 2.2.4, resulting for each camera c and pair $\langle \mathbf{x}^{cl}, \mathbf{x}^{cr} \rangle$ in the following system of homogeneous equations:

$$\begin{cases} \left[\tilde{\mathbf{x}}^{cl}\right]_\times \mathbf{K}_{cl} \left[\mathbf{R}^{cl}_{rig} \mid \mathbf{t}^{cr}_{rig}\right] \tilde{\mathbf{X}}^{rig} = \mathbf{0}, \\ \left[\tilde{\mathbf{x}}^{cr}\right]_\times \mathbf{K}_{cr} \left[\mathbf{R}^{cr}_{rig} \mid \mathbf{t}^{cr}_{rig}\right] \tilde{\mathbf{X}}^{rig} = \mathbf{0}. \end{cases} \tag{4.2}$$

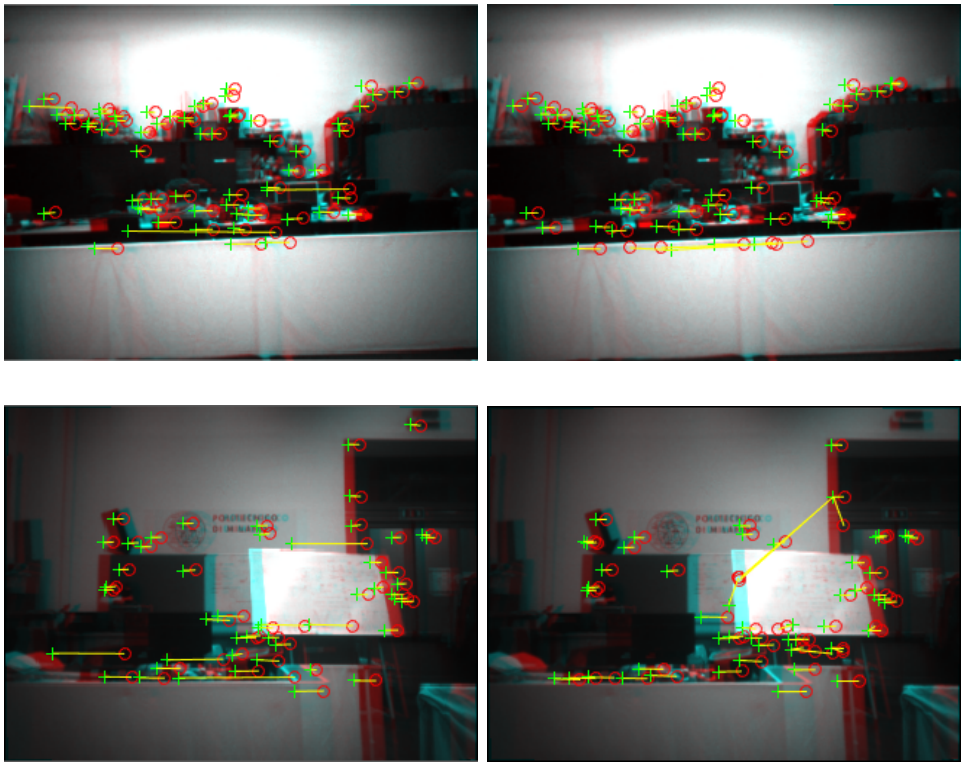
For weighting objective functions, as for instance in absolute orientation, we employ inverse keypoints depth in corresponding stereo frame, considering it as a approximation of their uncertainty. In such situation we perform linear triangulation in the stereo frame as follows:

$$\begin{cases} \left[\tilde{\mathbf{x}}^{cl}\right]_\times \mathbf{K}_{cl} \left[\mathbf{I} \mid \mathbf{0}\right] \tilde{\mathbf{X}}^{cl} = \mathbf{0}, \\ \left[\tilde{\mathbf{x}}^{cr}\right]_\times \mathbf{K}_{cr} \left[\mathbf{R}^{cr}_{cl} \mid \mathbf{t}^{cr}_{cl}\right] \tilde{\mathbf{X}}^{cl} = \mathbf{0}; \end{cases} \tag{4.3}$$

then, after the association of each 3D point with its depth (z coordinate), we bring them in the common rig frame by means of the transformation $\mathbf{R}^{rig}_{cl}$, $\mathbf{t}^{rig}_{cl}$. As a result of this process a sparse reconstruction of the observed scene from one perception is obtained, represented by a set of 3D points in a common reference frame each one associated with a visual descriptor (e.g. taken from the left originating keypoint), and eventually its depth in the corresponding stereo frame.

Algorithm 1 Epipolar Search Algorithm

```
1: procedure EPIPOLAR SEARCH
   Input: Left and right keypoints arrays kL and kR in a stereo pair of images
   Output: Set of matches M
2:     sort kL according to ≺, ≻, ≈ for row coordinates
3:     sort kR according to ≺, ≻, ≈ for row coordinates
4:     idxR = 0
5:     for idxL = 1 to kL.size() do
6:         if idxR == kR.size() then
7:             break
8:         while kL[idxL].row ≺ kR[idxR].row do
9:             idxL += 1                          ▷ Right on lower row, skip left
10:            if idxL == kL.size() then
11:                break
12:        while kL[idxL].row ≻ kR[idxR].row do
13:            idxR += 1                          ▷ Right on upper row, skip right
14:            if idxR == kR.size() then
15:                break
16:        idxRsearch = idxR
17:        bestDistance = maxDistance
18:        idxRbest = -1
19:        while kL[idxL].row ≈ kR[idxR].row do   ▷ Epipolar constraint
20:            if kR[idxRsearch].col > kL[idxL] then
21:                break                          ▷ Negative disparity
22:            if kR[idxRsearch].available then
23:                d = distance(kR[idxRsearch].desc, kL[idxL].desc)
24:                if d < bestDistance then
25:                    bestDistance = d
26:                    idxRbest = idxRsearch
27:            idxRsearch += 1
28:        if idxRbest ≠ -1 then                  ▷ Found a match
29:            M += ⟨kL[idxL], kR[idxRbest]⟩
30:            kR[idxRbest].available = false
31:            idxR = idxRbest + 1
```



(a) epipolar search *(b) brute force matching*

Figure 4.1: Two examples of stereo matching obtained by epipolar search (4.1a) and brute force (4.1b) over SURF features extracted from left and right images taken with a stereo camera. Each of the four pictures represents the overlap of left and right stereo images, in which left keypoints are represented by red circles and right keypoints by green crosses. Matched features are joined by a yellow line. The epipolar line thickness value used in epipolar search 2 pixels.

4.3 Matching

With the aim of inserting the localization algorithm in a stereo pipeline, we choose matching procedures able to find correspondences between a set of 3D points in reference frame w with associated descriptors and a multi-stereo perception. However, since in 3D3D approaches we need only to match points survived to stereo matching, and in 3D2D approaches we employ only left cameras perceptions in reprojection error minimization, we wish to match scene points with their images in left cameras.

The cases in which a rig pose guess is available in frame w or not are distinguished. In both cases matching can be made faster with respect a brute force approach by organizing elements in an indexing structure. In

the latter case however we can only rely on the visual similarity information given by descriptors, thus we employ classical matching procedures that divide the training descriptor set in fast accessible bins. Depending on the kind of descriptors, different algorithm can be used, as explained in Subsection 2.3.6 (e.g. SURF descriptors with FANN matching [12]).

Instead, when a pose guess is available we exploit geometrical information to speed up the matching as explained in Section 2.6, by projecting points and looking for correspondences in a squared patch around the projection. Depending on the situation we might want to project all 3D points on each camera or not. For a sensor with non overlapping FOVs, when keypoints are tracked between subsequent frames it is possible to project on each left camera only points that have been reconstructed in the previous frame using the corresponding stereo pair. This practically translates into the assumption that a scene point is not seen by two different cameras in two subsequent frames. When instead an already built map or a part of it needs to be matched with the current view, the temporal information we assumed before is not available, and we need to project points on each camera to check whether they fall inside the considered camera FOV. Note that this also applies to sensors with overlapping FOVs. We consider here the more general situation in which all points need to be projected on each camera.

The data structure we use resume the geometrical positioning of keypoints in images. For each camera we build a matrix with the same size of the corresponding image, and we place in position r, c the keypoint whose rounded image coordinates equal r, c, if this exists. The obtained structures can be thought as virtual images, containing in each keypoint coordinates its corresponding information. Then we stack together these virtual images in a tensor. At this point a guess projection can be obtained for each 3D point, giving a prediction of where the corresponding keypoint is placed in the tensor. The third dimension where to look simply depends on which camera projection matrix has been used for the prediction. The full procedure is shown in Algorithm 2.

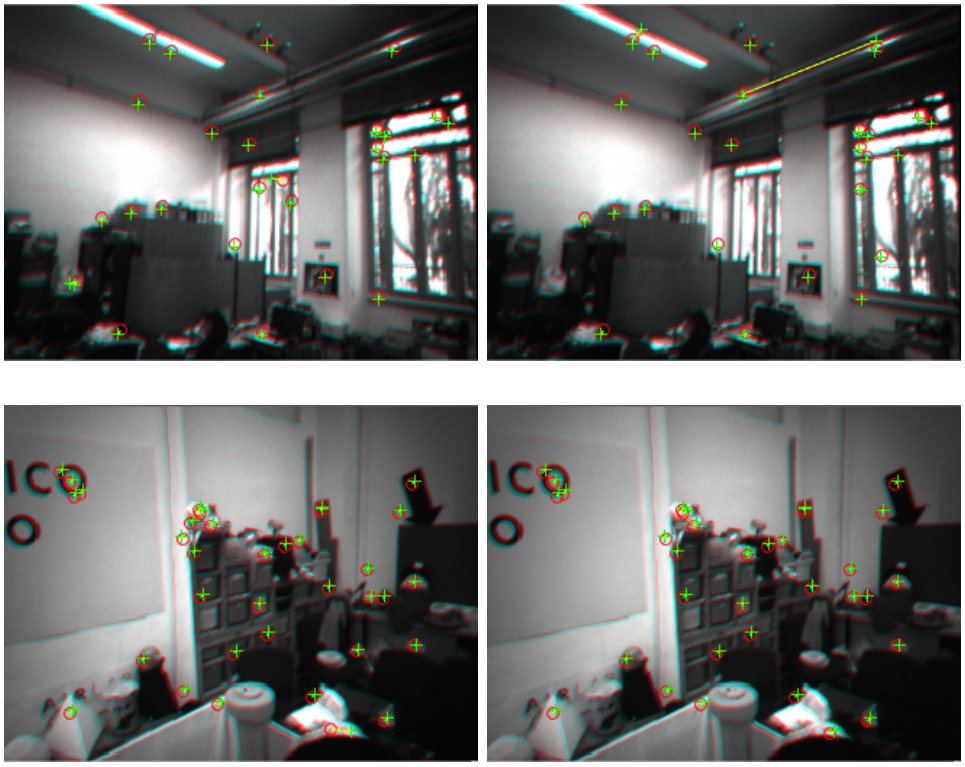
We show in Figure 4.2 a example of the applications of brute force matcher and the just described matching procedure for two back-to-back stereo cameras among two consecutive frames.

4.4 Localization

Now that the steps of reconstruction and matching are explained, we can pass to geometrical localization assuming a multi-stereo perception and a

Algorithm 2 Matching by projection

1: **procedure** MATCHING BY PROJECTION
Input: pts3D expressed in w, kL left keypoints, $\hat{\mathbf{R}}_w^{rig}$, $\hat{\mathbf{t}}_w^{rig}$ extrinsics guess
Output: Set of matches M
2: index = NULL[height][width][camerasNumber]
3: **for** k in kL **do** ▷ Fill the index with keypoints
4: row = round(k.row)
5: col = round(k.col)
6: index[row][col][k.cameraID] = k
7: **for** c = 1 to camerasNumber **do** ▷ Extrinsics guesses in w
8: $\hat{\mathbf{R}}_w^c = \mathbf{R}_{rig}^c \hat{\mathbf{R}}_w^{rig}$
9: $\hat{\mathbf{t}}_w^c = \mathbf{R}_{rig}^c \hat{\mathbf{t}}_w^{rig} + \mathbf{t}_{rig}^c$
10: **for** p3D in pts3D **do**
11: **for** c = 1 to camerasNumber **do**
12: p3Dcamera = eculidTransform(p3D, $\hat{\mathbf{R}}_w^c$, $\hat{\mathbf{t}}_w^c$)
13: **if** p3Dcamera.z ≤ 0 **then** ▷ Outside FOV, skip
14: **continue**
15: guess = $\pi(\mathbf{K}_{cl}, \mathbf{I}, \mathbf{0}, \text{p3Dcamera})$ ▷ Project on camera c
16: **if not** isInFOV(p3D, c) **then** ▷ Outside of FOV, skip
17: **continue**
18: ▷ Retrieve keypoints inside a patch around guess
19: tlRow = round(guess.row - patchSize/2)
20: tlCol = round(guess.col - patchSize/2)
21: brRow = round(guess.row + patchSize/2)
22: brCol = round(guess.col + patchSize/2)
23: candidates = index[tlRow : brRow][tlCol : brCol][c]
24: **if** candidates.isEmpty() **then**
25: **continue**
26: ▷ Look for best correspondence among candidates
27: bestDist = maxDist
28: bestCandidate = NULL
29: **for** kCandidate in candidates **do**
30: dist = distance(p3D.desc, kCandidate.desc)
31: **if** dist < bestDist **then**
32: bestDist = dist
33: bestCandidate = kCandidate
34: ▷ If something found add to matches
35: **if** bestCandidate **then**
36: M += $\langle$p3D, bestCandidate$\rangle$



(a) Reprojection matching *(b) Brute force matching*

Figure 4.2: Frame-to-frame matching obtained between two subsequent acquisitions of a multi-stereo sensor made of two back-to-back stereo cameras. A comparison of matching by projection under constant motion model (4.2a) and brute force matching (4.2b) is shown. Left images of two subsequent frames are superimposed, and keypoints at frame t are represented by red circles while keypoints at frame $t+1$ by green crosses. Obtained matches are joined by a yellow line. For matching by projection correspondences are searched in squared patches with side of 20 pixels.

sparse scene reconstruction to be given. We focus on 3D3D and 3D2D approaches for two main reasons:

- The scene reconstruction can be used to speed up the matching as previously explained.
- They can be easily employed in SFM and VSLAM applications.
- A single scene point can be associated to many perceptions, allowing to apply BA procedures for joint refinement over multiple views.

We start the 3D3D approach in Subsection 4.4.1, then we explain decoupled and coupled 3D2D solutions in Subsection 4.4.2. Finally we introduce in Subsection 4.4.3 an hybrid approach among the two.

4.4.1 3D3D

We saw in Section 2.5 that absolute orientation has closed form solution and requires a minimum of three correspondences to be solved. Since with multi-stereo sensors we are able to perform a scene reconstruction from a single view in the rig frame, the same 3D3D approach can be easily applied in these settings. We assume world to rig correspondences $\{\langle \mathbf{X}_i^w, \mathbf{X}_i^{rig} \rangle\}_{i=1..N}$ to be given, and we run the closed form solution based on quaternions of [18] inside RANSAC for outliers rejection. At each step, a candidate Euclidean transformation $\hat{\mathbf{R}}_w^{rig}$, $\hat{\mathbf{t}}_w^{rig}$ is computed from 3 sampled pairs, and the corresponding inliers set is formed by all the correspondences $\langle \mathbf{X}_i^w, \mathbf{X}_i^{rig} \rangle$ satisfying:

$$\left\| \hat{\mathbf{R}}_w^{rig} \mathbf{X}_i^w + \hat{\mathbf{t}}_w^{rig} - \mathbf{X}_i^{rig} \right\| \leq t, \tag{4.4}$$

where t is a fixed object space error threshold. When the best inliers set is found the transformation can be computed over all its elements via least squares. In this step we employ depth information associated to rig points as an approximation of their confidence, by weighting them with their inverse depth in the computation of centroid and cross covariance matrix.

4.4.2 3D2D

We implement two different 3D2D localization procedures providing the same interface. We assume to know a set of correspondences from 3D world points to left image points in the current perception, and we minimize the reprojection error over the best set of 3D2D pairs. Right points are not used here, however they can be included with no particular modifications to the algorithms. This approach does not need the configuration to be multi-stereo, because no world reconstruction is performed in the localization step, differently from the 3D3D approach. Thus the two localization procedures explained in this section also work for a multi-camera, if a scene reconstruction composed of 3D points matched with their current perception is given.

Note that for each image point the ID of the camera perceiving it is known. Our objective is to retrieve the principal camera extrinsics expression in the world frame. We distinguish two approaches: a decoupled one in which each left camera is independently localized and then the results are fused, and a coupled one in which all data are simultaneously used in the fitting of the rig pose.

The decoupled solution works as follows. We isolate for each stereo camera c all the correspondences $\langle \mathbf{X}^w, \mathbf{x}^{cl} \rangle$ and we remove wrong correspon-

dences by running P3P inside RANSAC, obtaining the extrinsics $\hat{\mathbf{R}}_w^c$, $\hat{\mathbf{t}}_w^c$ with biggest consensus set $\mathcal{I}_c$. We iteratively refine this solution by minimizing the following error function over all inlier correspondences:

$$\mathbf{r}_w^c, \mathbf{t}_w^c = \underset{\mathbf{r},\mathbf{t}}{\operatorname{argmin}} \sum_{\langle \mathbf{X}^w, \mathbf{x}^{cl} \rangle \in \mathcal{I}_c} \|\mathbf{x}^{cl} - \pi\left(\mathbf{K}_{cl}, \mathcal{R}(\mathbf{r}), \mathbf{t}, \mathbf{X}^w\right)\|^2, \tag{4.5}$$

where $\mathbf{r}_w^c$ is the Rodrigues representation of $\mathbf{R}_w^c$, and $\mathcal{R}(\cdot)$ is the operator bringing the minimal Rodrigues representation in its corresponding rotation matrix. For solving the problem we employ the Levenberg Marquardt optimization algorithm, and the obtained camera extrinsics are translated in principal camera extrinsics estimation thanks to calibration:

$$\begin{aligned} {}^c\mathbf{R}_w^{rig} &= \mathbf{R}_c^{rig}\,\mathbf{R}_w^c, \\ {}^c\mathbf{t}_w^{rig} &= \mathbf{R}_c^{rig}\,\mathbf{t}_w^c + \mathbf{t}_c^{rig}. \end{aligned} \tag{4.6}$$

The global estimation of principal camera extrinsics is obtained by averaging, i.e., the translation component is the arithmetic mean of different translation estimations in $\{{}^c\mathbf{t}_w^{rig}\}_{c=1..C}$, while for the rotation we compute the geodesic mean over $\{{}^c\mathbf{R}_w^{rig}\}_{c=1..C}$, being single estimations elements of a manifold. A simple algorithm computing the geodesic mean of different rotations is described in [57]; it iterates mean computation in the tangent space and projection on the manifold. We approximate different confidences in pose estimations weighting elements by inliers number in each camera divided by the average depth of inliers points, if this is available.

A coupled solution can be obtained by considering the generalized imaging model, allowing for many projection centers. In these settings we have a single RANSAC run, with as input the set of all 3D2D correspondences. At each iteration 4 random pairs are sampled, and 3 of them are employed in the solution of the following equation system for the principal camera extrinsics $\hat{\mathbf{R}}_w^{rig}$, $\hat{\mathbf{t}}_w^{rig}$:

$$\begin{cases} \pi_{c_1 l}(\hat{\mathbf{R}}_w^{rig}, \hat{\mathbf{t}}_w^{rig}, \mathbf{X}_i^w) = \mathbf{x}_i^{c_1 l}, \\ \pi_{c_2 l}(\hat{\mathbf{R}}_w^{rig}, \hat{\mathbf{t}}_w^{rig}, \mathbf{X}_j^w) = \mathbf{x}_j^{c_2 l}, \\ \pi_{c_3 l}(\hat{\mathbf{R}}_w^{rig}, \hat{\mathbf{t}}_w^{rig}, \mathbf{X}_k^w) = \mathbf{x}_k^{c_3 l}; \end{cases} \tag{4.7}$$

a set of 8 possible solutions is computed using the gP3P method of [50]. A unique solution is then obtained by choosing the one giving smallest reprojection error for the fourth sampled point. The principal camera extrinsics is finally obtained by reprojection error minimization over all the elements of the biggest consensus set $\mathcal{I}$, containing 3D2D correspondences potentially

from each camera:

$$\mathbf{r}_w^{rig}, \mathbf{t}_w^{rig} = \underset{\mathbf{r},\mathbf{t}}{\operatorname{argmin}} \sum_{\langle \mathbf{X}^w, \mathbf{x}^{cl} \rangle \in \mathcal{I}} \left\| \mathbf{x}^{cl} - \pi_{cl} \left(\mathcal{R}(\mathbf{r}), \mathbf{t}, \mathbf{X}^w \right) \right\|^2, \tag{4.8}$$

for wich the principal camera extrinsics with bigger consensus set is used as starting solution, and iteratively refined using the Levenberg Marquardt optimization algorithm.

4.4.3 Hybrid

A different approach is to consider reprojection errors inside the outliers rejection scheme by computing candidate principal camera extrinsics via point clouds alignment. Here we assume a set of triples $\{\langle \mathbf{X}_i^w, \mathbf{X}_i^{rig}, \mathbf{x}_i^{cl} \rangle\}_{i=1..N}$ to be given, where each one contains a world point, its corresponding reconstruction in rig frame, and its left image on some camera c. As in the classical 3D3D method 3 correspondences are sampled at a time and absolute orientation is solved for $\hat{\mathbf{R}}_w^{rig}$, $\hat{\mathbf{t}}_w^{rig}$. The inlier set is now composed of all the 3D3D correspondences satisfying:

$$\left\| \mathbf{x}_i^{cl} - \pi_{cl} \left(\hat{\mathbf{R}}_w^{rig}, \hat{\mathbf{t}}_w^{rig}, \mathbf{X}_i^w \right) \right\| \leq t, \tag{4.9}$$

where t is a fixed threshold in pixels. Note that differently from the classical 3D3D approach this method exploits the information of camera ID associated to each keypoint of the current perception, needed to choose the camera on which wolrd points should be projected. When the consensus set is found it is possible to choose between point clouds alignment and reprojection error minimization to find the final solution.

A quality of this method is that as the 3D3D approach straightforwardly extends to multi-stereo sensors, and lets to use points coming from all cameras jointly. However it resembles the depth error attenuation of 3D2D methods in the outliers rejection scheme. A comparison example can be seen in Figure 4.3, where both methods are applied on subsequent perceptions of a single stereo camera, and resulting inliers correspondences are shown. The 2D error method is clearly able to include further points in the inliers set than the object space error method. However these should be used with attention in pose estimation, for instance by adding them in a map only when supported by many views as in ORB SLAM 2 [28] or disabling their translational contribution as done in ProSLAM [14].

(a) Reprojection error inliers

(b) Object space error inliers

Figure 4.3: Visual comparison of reprojection error (4.3a) and object space error (4.3b) metrics for outliers rejection by point clouds alignment on a stereo camera. Consecutive left images are superimposed, and keypoints at frame t are represented by red circles while keypoints at frame $t+1$ by green crosses. Obtained matches are joined by a yellow line.

4.5 Case Study: Visual Odometry

In this section we introduce a simple visual odometry procedure based on SFM to show that the developed localization algorithms can be employed in

classical stereo pipelines. Indeed most stereo VO and VSLAM algorithms make the basic assumption that the scene can be reconstructed from a single view, and a set of matches between the current perception and a sparse map of 3D points can be found. These two assumptions have been respectively satisfied in Sections 4.2 and 4.3 for a calibrated multi-stereo sensor.

We compute the principal camera pose incrementally by composition of frame-to-frame relative transformations, from matches between the current perception and the scene reconstruction computed in previous view. Matches are found using Algorithm 2 under constant motion assumption. This approach is simpler than the map-to-frame approach, in which a set of points belonging to an already built map is used for localization, however this more general case could still be applied in our settings because our matching procedure does not make assumptions on the input point cloud.

The full VO pipeline is shown in Figure 4.4. At each time-stamp 3D scene points are computed in the principal camera frame using the reconstruction algorithm explained in Section 4.2. In 3D3D approaches we align these points with the previous frame reconstruction, while in 3D2D approaches only corresponding left camera points are used either in coupled or decoupled reprojection error minimization. The obtained roto-translation is then composed with previous extrinsics estimation in world frame, resulting in the current extrinsics estimation. The described step gives for each perception the principal camera extrinsics in world frame, that we place on the initial camera pose, and a set of 3D2D inlier correspondences. We store this information in a map of points, where each 3D point is associated with many of its perceptions at different estimated poses, consisting in left pixels coordinates and ID of the perceiving camera. We call point track the list of perceptions associated to a 3D point.

For incrementally building the map the set of 3D points in the current reconstruction is split in two: tracked ones, resulted as inliers in relative pose estimation, and non-tracked ones, that either have not been matched or have been matched but resulted as outliers. Since tracked points are already present in the map, we add to their point track their current perception. Non-tracked points are instead transformed according to the current camera pose in such a way they are expressed in world frame, and then added to the map together with their current perception. Note that due to its semplicity the frame-to-frame approach does not handle the situation in which a scene point is lost in one view, differently from the map-to-frame approach. Indeed as soon as a scene point does not appear as inlier in relative pose estimation, its point track stops to grow. If the scene point is perceived again the algorithm is unaware of it, and a new map point is added to the map. In

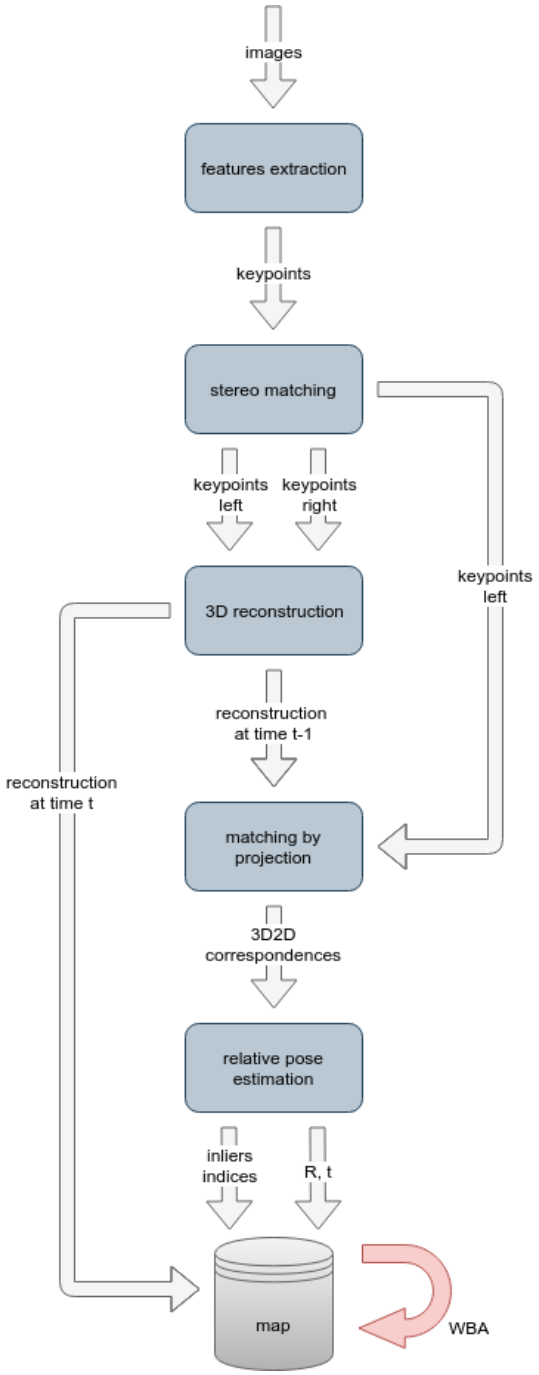


Figure 4.4: Our VO pipeline. After reconstruction, left image points are matched with previous reconstruction for incremental motion estimation. 3D points are then added to the map, together with their perceptions. Windowed bundle adjustment is then run over map data.

many state of the art procedures this is solved by projecting the map on current images to retrieve matches that have been previously lost.

The built map contains all the information needed to perform joint reprojection error minimization. To keep reduced computational cost optimization is applied only on a fixed size window of time in pose only windowed bundle adjustment fashion, refining only the last N poses:

$$\min_{\mathbf{r}_t,\mathbf{t}_t} \sum_t \sum_c \sum_i \|\mathbf{x}_{i,t}^c - \pi_c(\mathcal{R}(\mathbf{r}_t), \mathbf{t}_t, \mathbf{X}_i)\|^2, \tag{4.10}$$

where $\mathbf{r}_t$, $\mathbf{t}_t$ are camera extrinsics in world frame at time t, $\mathbf{X}_i$ a map point, and $\mathbf{x}_{i,t}^c$ an image taken by camera c at time t of point $\mathbf{X}_i$. As usual the problem is solved using Levenberg Marquardt.

4.6 Calibration

The calibration of a multi-camera sensor consists in estimating intrinsic parameters of each camera, plus their extrinsics in the rig frame. Potentially all state of the art calibration procedures explained in Section 3.3 can be applied to multi-stereo sensors by considering only left cameras of each pair, and calibrating independently each stereo camera. However a better approach would be to first calibrate each stereo camera with classical photogrammetric calibration procedures, known to provide very accurate results, and then exploit their knowledge in the estimation of left cameras poses in rig frame. In this way self-calibration procedures (both hand-eye and VSLAM based) do not require any more an initialization phase in which map and trajectory scales are recovered, since SFM pipelines including stereo sensors can directly estimated absolute trajectory and map. Moreover, when some sort of reprojection error minimization is employed, also right camera perceptions can be included increasing the number of constraints the solution should satisfy, both in self-calibration and photogrammetric approaches.

Since the sensor on which we will test the localization algorithms gives the possibility of simultaneously observing some different part of a planar calibration pattern with different cameras, we choose to adapt the calibration procedure of [56] to a multi-stereo sensor. This sacrifices the generality of the approach (it is not for instance applicable to two back-to-back facing stereo cameras), however allows to use a single calibration pattern and results in an easy to implement calibration procedure. The choice of following a photogrammetric approach is based on the fact that, in self-calibration procedures, drift in motion estimation could happen degrading the calibration accuracy.

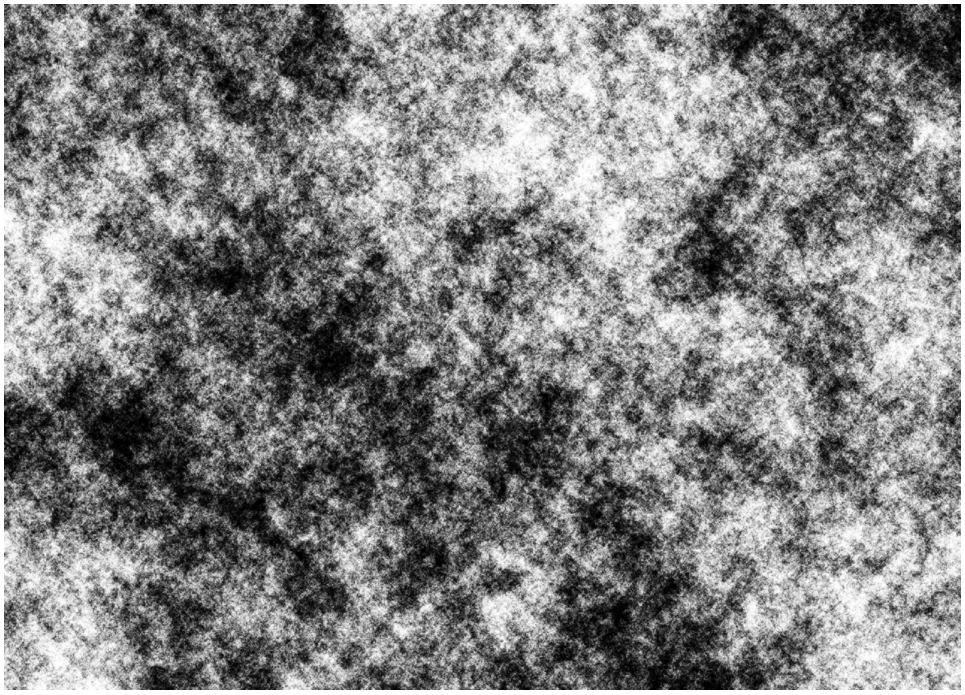


Figure 4.5: The used planar calibration pattern developed in [56] by reverse engineering of SURF extractor

The used calibration pattern is shown in Figure 4.5. This has been designed by the authors to provide an high number of SURF features, simplifying the task of localizing it in the frame of a calibrated camera even when not completely visible. The localization procedure works as described in Subsection 2.4.2: keypoints are extracted from both a template image of the pattern and perceived image, then matches are obtained and outliers are removed by homography fitting inside RANSAC. Since the size of the pattern is known, template keypoints can be scaled in such a way they reflect the real pattern size. As result the obtained homography contains projection matrix elements in pattern frame, from which extrinsics can be recovered as explained for single camera calibration in Subsection 2.1.3. Similarly PnP can be solved instead of homography fitting.

The authors of [56] developed a multi-camera calibration algorithm computing both intrinsics and extrinsics of each camera, based on planar object localization and bundle adjustment. When intrinsics refinement is required pattern points are optimized with BA, while when only extrinsics fitting is required a motion only BA procedure is employed. We modify the original algorithm adding the following modifications to adapt it to our settings:

- We first calibrate each stereo camera with classical photogrammetric procedure using a checkerboard pattern, known to provide accurate results. This is useful for introducing additional knowledge in the multi-camera calibration, plus assure accurate image rectification and triangulation in the localization scenario.

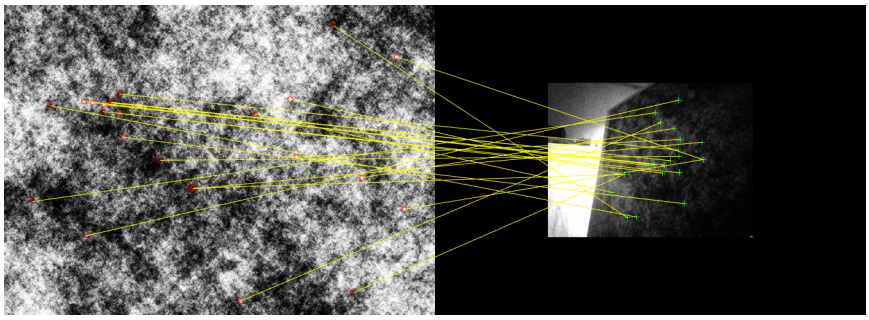
- We substitute SURF with its affine invariant version FAIR-SURF [58].

- We perform motion only BA with the possibility of optimizing a scale factor to handle imprecision in pattern size measure.

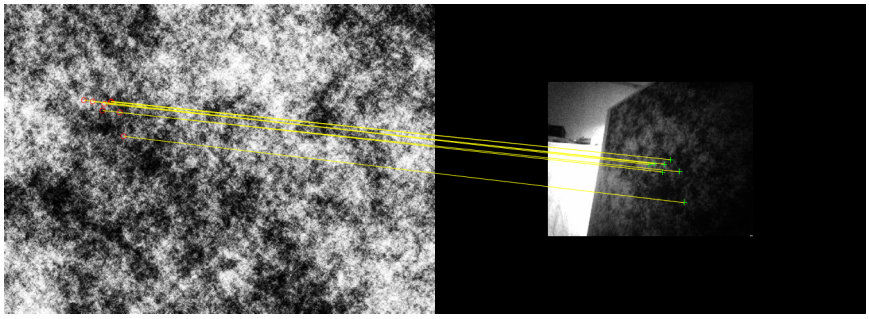
For non overlapping FOVs sensors, when images are taken in such a way two different cameras observes the pattern simultaneously, an high affine effect is often present. In this situation it may happens that SURF does not perform well, and the outliers rejection procedure fails. FAIR-SURF (Fully Affine InvaRiant SURF) solves this problem by applying the IMAS (Image Matching by Affine Simulation) technique. This consists in computing a pre-determined set of affine transformations to the image, in order to simulate the transformations induced by the viewpoint changes. Note that this approach is very similar to the scale invariant technique introduced in SIFT, where different scales are simulated by blurring and keypoints are detected at different scales. We show a comparison of SURF and FAIR-SURF performance on a calibration image in Figure 4.6.

The inputs of the calibration procedure are the individual stereo cameras calibration, and a set of stereo images obtained while observing the pattern, each one associated with a time instant. Note that only images of the pattern taken by different stereo cameras at the same time give information about their relative position. The calibration procedure can be divided in two fundamental steps: the first one in which images are processed and a pose graph is built, and a second one in which the information stored in the graph are used for obtaining a calibration guess that is then refined in BA fashion.

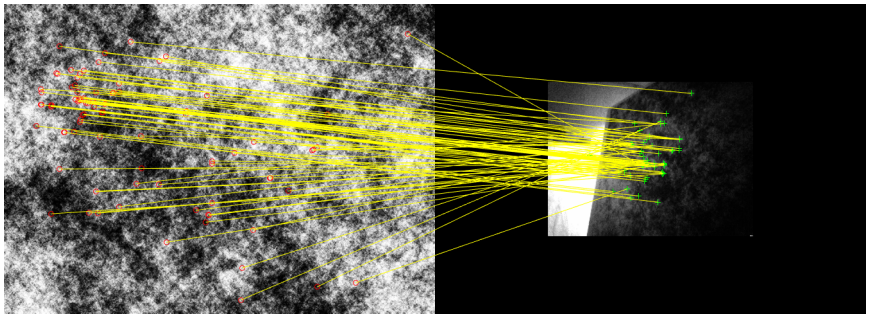
A pose graph is composed of one node for each camera, and one node for each pattern pose at some given time instant. However in our extension to multi-stereo cameras we assign a single node to each stereo sensor, since left and right images are used for pattern localization in the stereo camera frame, and the transformation relating left and right cameras is never modified. We call respectively c_i and p_t the nodes corresponding to stereo camera i and pattern pose at time t. The only allowed connections are between pattern and camera nodes, and an edge connects c_i and p_t if the stereo camera i observes the pattern at time t. This edge stores information obtained by



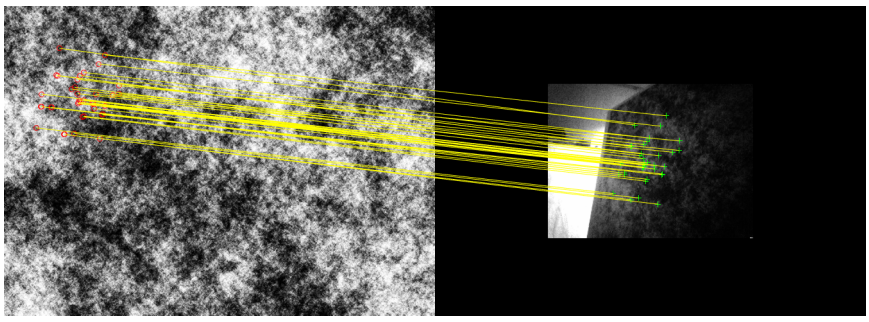
(a) Matched SURF features



(b) Inliers among matched SURF features

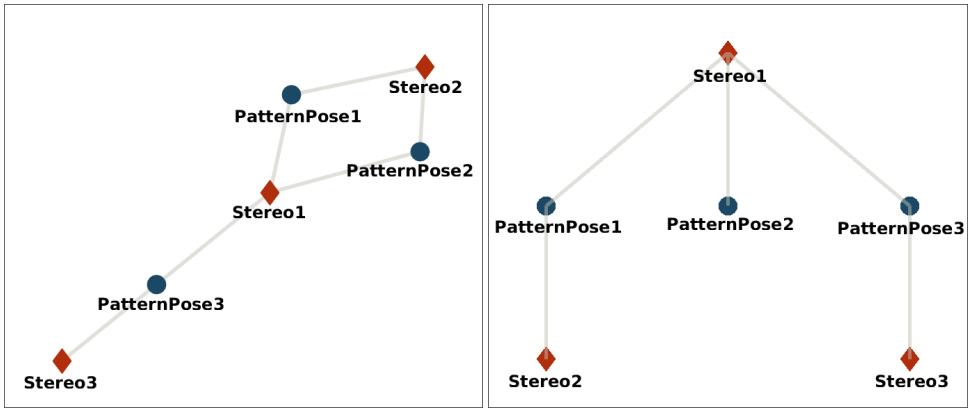


(c) Matched FAIR-SURF features



(d) Inliers among matched FAIR-SURF features

Figure 4.6: Visual comparison of matches between a template image of the pattern and a sample calibration image, obtained with SURF and FAIR-SURF.



(a) Pose graph *(b) Spanning tree*

Figure 4.7: Pose graph (4.7a) of a multi-camera composed of three stereo pairs, taking pictures of the pattern in three different time instant along which the pattern is moved. Stereo cameras are represented by red diamonds and pattern poses by blue circles. In its first two poses the pattern falls in the FOVs of stereo camera 1 and stereo camera 2, while in the third the pattern is observed by stereo camera 1 and stereo camera 3. In (4.7b) we represent the corresponding spanning tree, with root in the principal camera (Stereo1).

the planar object localization procedure, consisting in pattern to left and pattern to right inliers correspondences, and the relative position of the stereo camera i with respect to the pattern pose at time t. An example of a pose graph for calibrating a rig composed of three stereo sensors over three pattern images acquisition is shown in Figure 4.7.

We now explain the pattern localization procedure, operating on a stereo pair of images of the pattern. FAIR-SURF keypoints are extracted from both images and matched by FANN with pattern keypoints. We decide here to reject outliers independently on left and right image, since it often happens that the matching procedure gives bad correspondences in only one of the two images. A joint approach could lead in this situation to and high outliers percentage, even when one of the images gives good matches. We choose to apply p3p instead of homography fitting in outliers rejection, since it has shown to perform better in our experiments. At this point images with a number of inliers lower than a fixed threshold are discarded, and PnP is applied over points coming from the image with higher number of inliers. If both left and right images obtained enough inliers, reprojection error is jointly minimized to increase coupling:

$$\min_{\mathbf{r},\mathbf{t}} \sum_{\langle \mathbf{X},\mathbf{x} \rangle \in \mathcal{I}_l} \|\mathbf{x} - \pi_l(\mathcal{R}(\mathbf{r}), \mathbf{t}, \mathbf{X})\|^2 + \sum_{\langle \mathbf{X},\mathbf{x} \rangle \in \mathcal{I}_r} \|\mathbf{x} - \pi_r(\mathcal{R}(\mathbf{r}), \mathbf{t}, \mathbf{X})\|^2, \quad (4.11)$$

where $\mathcal{I}_l$ and $\mathcal{I}_r$ are respectively the inliers sets found in left and right images, and the functions $\pi_l(\cdot)$ and $\pi_r(\cdot)$ compute the projection of a 3D point on left and right cameras:

$$\begin{aligned}\pi_l(\mathbf{R}, \mathbf{t}, \mathbf{X}) &= \pi(\mathbf{K}_l, \mathbf{R}, \mathbf{t}, \mathbf{X}),\\ \pi_r(\mathbf{R}, \mathbf{t}, \mathbf{X}) &= \pi(\mathbf{K}_r, \mathbf{R}_l^r\,\mathbf{R}, \mathbf{R}_l^r\,\mathbf{t} + \mathbf{t}_l^r).\end{aligned}$$

When all images are processed the complete pose graph is built, containing one node for each pattern pose connected to observing camera nodes. Pattern nodes that are not connected to at least two camera nodes are discarded, since they do not provide information for extrinsics calibration.

As suggested by the authors of [56], in the optimization phase all poses are expressed in a common frame, chosen to be the one of the principal camera. To this extent a spanning three with root in the principal camera node is obtained, and traversed from the root c_1 to each leaf. To each node n we assign a position obtained by composition of transformations stored on edges forming the path from c_1 to n, that we call $\mathbf{R}_n$, $\mathbf{t}_n$. Note that transformations stored in edges traversed from a pattern node p to a camera node c need to be inverted before composition, since these represent the extrinsics of c in pattern pose p.

This procedure gives a guess for the non-linear optimization problem of finding all pattern poses and camera poses that minimize the sum of squared reprojection error over all inlier points.

$$\min \sum_{\mathcal{C}} \sum_{\mathcal{P}} \left(\sum_{\mathcal{I}_{cl}^p} \|\mathbf{x} - \pi_{cl}(\mathbf{R}_p^c, \mathbf{t}_p^c, \mathbf{X})\|^2 + \sum_{\mathcal{I}_{cr}^p} \|\mathbf{x} - \pi_{cr}(\mathbf{R}_p^c, \mathbf{t}_p^c, \mathbf{X})\|^2 \right) \tag{4.12}$$

where $\mathcal{C}$ is the set of stereo cameras, $\mathcal{P}$ the set of pattern poses, $\mathcal{I}_{cl}^p$ and $\mathcal{I}_{cr}^p$ respectively the inliers sets of left and right cameras belonging to stereo c when observing the pattern in pose p, and:

$$\begin{aligned}\mathbf{R}_p^c &= \mathbf{R}_c^T \mathbf{R}_p,\\ \mathbf{t}_p^c &= \mathbf{R}_c^T \mathbf{t}_p - \mathbf{R}_c^T \mathbf{t}_p.\end{aligned} \tag{4.13}$$

The minimization problem is solved using the Levenberg Marquardt optimization algorithm and rotations are expressed with Rodrigues representation, even if not explicit in Equation 4.12. Plus instead of inverting camera poses in the error function as shown in Equation 4.13, we simply keep their inverse (extrinsics in rig frame) as optimized variables. When also the scale factor is optimized 3D points in Equation 4.12 are multiplied by the optimized variable s.

4.7 Implementation Details

Since our objective is to compare different relative pose estimation algorithms for a multi-camera sensor, we choose MATLAB as implementation language. The obvious drawback is that this implementation is not directly portable in a real robotics scenario, however out of the box linear algebra, non-linear optimization, and computer vision functionalities allow to fast prototype different algorithms while keeping the focus on geometry and computer vision pipelines. Another important motivation is that the MATLAB robotics toolbox allows to easily access ROS (Robot Operating System) data, generally used for sensors interfacing. We now explain for completeness some details about our implementation.

In all algorithms keypoints extraction and matching is present. For keypoints extraction we use the MATLAB SURF implementation, and we implement FAIR-SURF in our multi-stereo calibration procedure by running this extractor on a predetermined set of affine transformations applied to the input image. Each time we perform keypoints matching without considering geometrical knowledge of the situation we use the MATLAB implementation of FANN on descriptors, while for stereo matching and matching from known pose we provide our custom implementations, very similar to the pseudo code provided in Algorithm 1 and Algorithm 2.

Another common problem is reprojection error minimization from a given pose guess. To this extent MATLAB code computing reprojection error and its Jacobian is automatically generated using the MATLAB symbolic toolbox, as functions taking in input camera pose (Rodrigues rotation representation and translation), calibration matrix, and 3D2D correspondences. This function is then fed to the MATLAB implementation of Levenberg-Marquardt algorithm that iteratively refines the given pose guess. An analogous approach has been used for optimization of the multi-stereo calibration and visual odometry objective functions.

The multi-stereo calibration procedure is developed in a class that represents the pose graph as a MATLAB graph, and provides methods for its construction from a set of stereo images and poses optimization. Relative positions between pattern and cameras are computed by using MATLAB p3p and RANSAC implementations, and joint reprojection error minimization on left and right stereo images is implemented as explained before.

We represent a multi-stereo camera with a class storing calibration data and providing methods giving Euclidean transforms between principal camera and other cameras of the sensors and vice versa. Cameras are identified with ordered numerical IDs. We represent a multi-stereo perception as key-

points in pixel coordinates and corresponding descriptors from left cameras, associated with numerical ID of the perceiving camera. The multi-stereo reconstruction procedure is based on MATLAB implementation of the linear triangulation method solving the homogeneous equation system with SVD, and returns 3D points in principal camera frame associated with descriptors and eventually depth data.

All relative pose estimation algorithms take in input a multi-stereo perception and a multi-stereo reconstruction, together with a multi-stereo camera instance from which calibration data are retrieved. The MATLAB RANSAC implementation is used in all of them. In 3D3D approaches we use our implementation of [18] for sparse reconstructions alignment, while in 3D2D decoupled approach we use MATLAB p3p implementation with our reprojection error minimization procedure. The 3D2D coupled approach uses inside the outliers rejection scheme the OpenGV[1] implementation of gP3P described in [50], and performs reprojection error minimization in the usual way.

In our visual odometry implementation we represent the map as a class storing a set of landmarks and extrinsics in wold frame, associated with timestamps. A landmark represents a 3D point, together with the corresponding point track. Point tracks are arrays of perceptions, where a perception is a left image point associated with corresponding camera ID and timestamp at which it has been perceived. This data organization makes simple to perform WBA over last n timestamps, by retrieving all points with some perception inside the considered window and last n stored extrinsics.

[1] http://laurentkneip.github.io/opengv

Chapter 5

Experimental Results

We describe in this chapter experiments and corresponding results on real data acquired in laboratory. In Section 5.1 we give a brief description of the used sensor. In Section 5.2 we show multi-stereo calibration results. In Section 5.3 we test the relative pose estimation algorithms and finally in Section 5.4 we show the results of our simple VO procedure on a short path.

5.1 Sensor

We perform experiments on the DJI Guidance sensor[1], a system of 5 stereo cameras 4 of which are arranged as the side of a square and the last points to the bottom. All cameras are synchronized, and takes images of size 320×240 pixels at 10 Hz. A picture of the sensor can be seen in Figure 5.1. For simplicity we will not use the bottom camera in our experiments, however this can be included with no modification of the developed algorithms.

Figure 5.1: The sensor used in our experiments.

[1] https://www.dji.com/it/guidance

Figure 5.2: An example of a stereo pair of calibration images after rectification.

5.2 Calibration

As explained in Section 4.6 we first independently calibrate each stereo sensor to obtain distortion parameters, calibration matrices, and left to right Euclidean transformations. For each stereo sensor we take about 40 images of a checkerboard with 6×8 corners and square size of 108 millimeters, that are then fed to the MATLAB stereo camera calibration toolbox based on [1]. Images are taken in such a way the whole image plane is covered, at various angulations and distances (about 1 to 7 meters). We report in Figure 5.2 a calibration image example, and resulting mean reprojection error over checkerboard corners in Table 5.1.

For finding left cameras extrinsic parameters in a common frame we apply our multi-stereo calibration procedure described in Section 4.6, placing the rig reference frame on stereo camera 1. We take about 40 images for each adjacent pair of stereo sensors in such a way they simultaneously observe part of the pattern, that has been generated using the code provided by the authors of [56] and printed on a rigid surface. The size of the pattern is 815×1136 millimeters. We report in Figure 5.3 two stereo pair of images employed in the calibration of our sensor. In the calibration procedure only images providing at least 20 inliers are kept, and a threshold of 1 pixel is used for outliers rejection with p3p inside RANSAC. These are then

	Stereo 1	Stereo 2	Stereo 3	Stereo 4
Images	40	38	38	41
Error mean (pixels)	0.0674	0.0483	0.0505	0.0562

Table 5.1: Number of images and reprojection errors in calibration with checkerboard of each stereo camera.

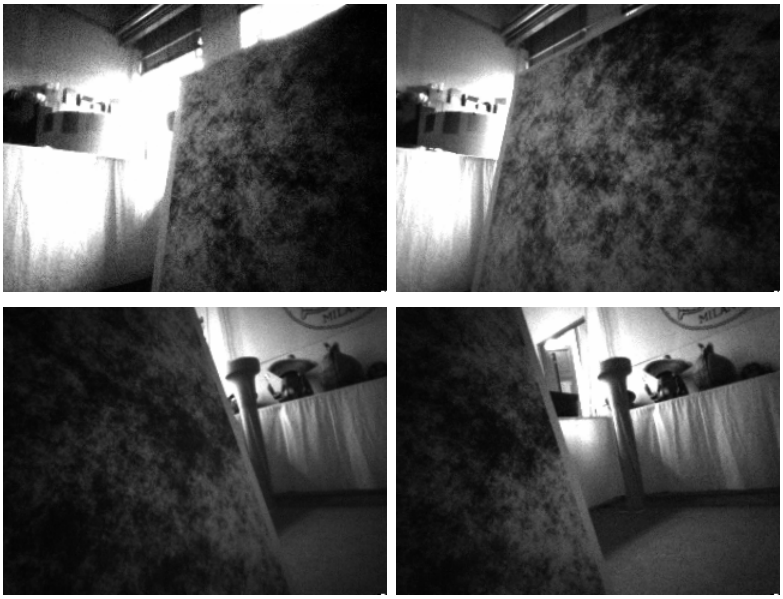


Figure 5.3: An example of two stereo pair of images of the pattern simultaneously taken by stereo camera 1 (top) and stereo camera 2 (bottom).

arranged in the pose graph shown in figure 5.4. We show in Table 5.2 the number of images and inlier points, together with reprojection error mean and standard deviation in the obtained solution. Thanks to the use of FAIR-SURF a big number of points is used, however the reprojection error is an order of magnitude bigger that the one obtained on each stereo sensor with the checkerboard calibration pattern. This is not only due to more difficult settings, but also to the fact that checkerboard corner detection has bigger subpixel accuracy than SURF. However the feature based approach of this calibration algorithm makes simple to detect and localize the pattern even when not fully visible, differently than classical checkerboard calibration. For completeness we show in Figure 5.5 the reprojection error distribution over each stereo camera.

	Images	Inliers	Error mean (pixels)	Error deviation (pixels)
Stereo 1	39	2514	0.5469	0.2403
Stereo 2	32	1792	0.5517	0.2446
Stereo 3	52	4102	0.5031	0.2376
Stereo 4	59	2807	0.5295	0.2425

Table 5.2: Number of images, inlier points, and reprojection errors mean and standard deviation in multi-stereo calibration applied to our sensor.

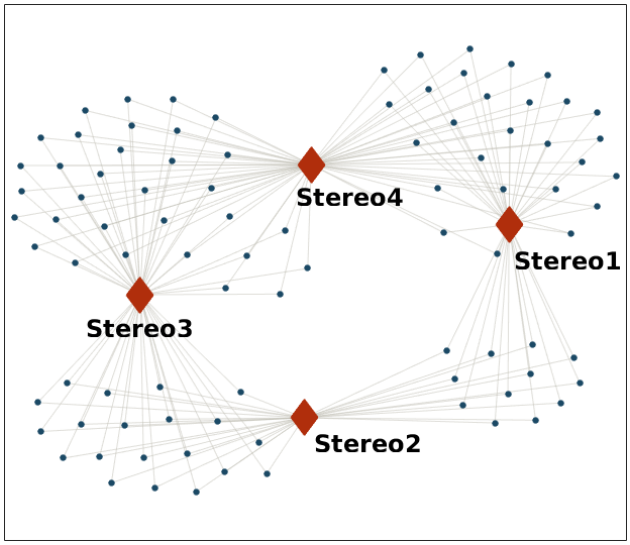
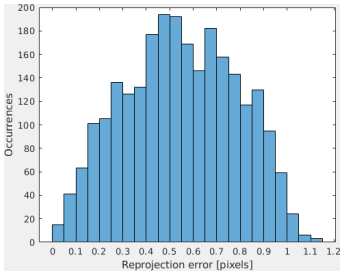
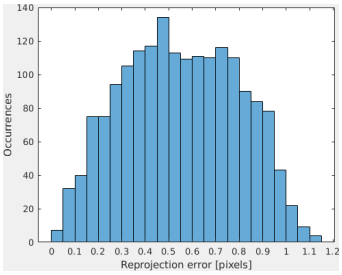


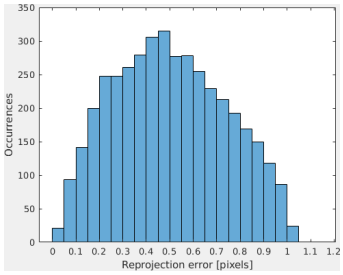
Figure 5.4: The pose graph obtained in the calibration of our multi-stereo sensor. Stereo camera nodes are represented as red diamonds and pattern pose nodes as blue circles. A pattern pose node is connected to a stereo camera node if the pattern falls inside its FOV. Note that the camera placement in our multi-stereo sensor allows to observe the pattern with only two stereo cameras simultaneously.



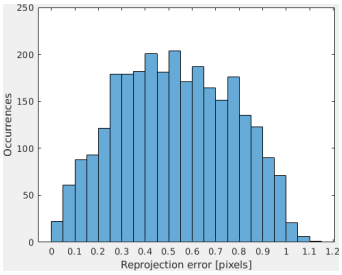
(a) Stereo camera 1.



(b) Stereo camera 2.



(c) Stereo camera 3.



(d) Stereo camera 4.

Figure 5.5: Resulting reprojection errors of the calibration procedure applied to our system of 4 stereo cameras. Each histograms includes both left and right points. Horizontal axis scale is in pixels while vertical represents the number of points falling in corresponding bin. Bins sizes are of 0.05 pixels.

5.3 Relative Pose Estimation

In this experiment we asses the capabilities of relative pose estimation algorithms described in Section 4.4. To this extent we capture some multi-stereo perceptions of the same scene from different points of view, and we apply relative pose estimation algorithms that are then compared with ground truth data.

5.3.1 Ground Truth

During image capturing a checkerboard pattern is positioned in the environment and kept fixed. In each acquisition the checkerboard is in the field of view of one sensor, and the rig is localized in the checkerboard frame. To this extent we look for a checkerboard pattern in each image, and we localize it in the frame of the perceiving stereo camera. Using extrinsics calibration we localize the principal camera in the checkerboard frame. The transformation relating two different camera positions is then computed by composition of camera to checkerboard transformations. The input images are then modified by the checkerboard localization procedure, by making black all pixels belonging to the checkerboard image. In this way the relative pose estimation algorithms do not make use of checkerboard points.

This is a very simple way of retrieving ground truth data, since, differently from external visual positioning systems, it directly gives an estimation of the principal camera triad, positioned in its focal point and oriented with the usual convention.

5.3.2 Experiment and Results

We take 10 perceptions of the same scene, where the first is the base perceptios and the subsequent are enumerated from 1 to 9. Poses from 1 to 4 are obtained by incrementally translating the sensor by 30 centimeters, giving a maximum displacement of 120 centimeters. Pose 5 is obtained by rotating the sensor in place of about 180 degrees from the base pose, pose 6 by performing the same in place rotation from pose 2, and so on up to pose 9 that corresponds to a translation of 120 centimeters plus a rotation of 180 degrees from the base pose. We extract SURF features from acquired images, then we employ our reconstruction procedure on the base perception. For each subsequent perception we apply FLANN matching to obtain correspondences with the scene reconstruction expressed in base frame, and we run the following algorithms:

- 3D2D: coupled 3D2D method based on gP3P and joint reprojection error minimization. A threshold of 6 pixels is employed in the outliers rejection scheme.

- Hybrid: the hybrid method performing point cloud alignment inside the outliers rejection scheme with reprojection error check for choosing inliers. Reprojection error minimization is applied over the inlier set for guess refinement. The pixel threshold is set to 6 also in this case.

- 3D3D: classical point cloud alignment, with an object space error threshold of 10 centimeters.

- dec3D2D: decoupled 3D2D method, computing each camera pose with p3p and reprojection error minimization, followed by weighted averaging. The pixel threshold is set to 6 pixels.

- Best Stereo: We independently run p3p followed by LHM (the globally convergent iterative PnP method we explained in Section 2.5.3) on data coming from each stereo sensor, then we use extrinsic calibration to obtain an estimation of the rig pose. We compare each camera result with ground truth and we choose the one with smaller translation error. This algorithm is not applicable in real settings since makes use of ground truth, however, it gives an idea of the best localization result we can obtain using a single camera over the reconstruction obtained from base images.

We run each algorithm on each pose 20 times and we average the localization errors and inliers number.

As shown in Figure 5.6, and Figure 5.7, the coupled 3D2D method outperforms other methods in almost all poses in terms of localization accuracy. The 3D3D method has a clear decrease of performance from pose 5 to 9, where the sensor has been rotated and each camera observes a part of the scene previously reconstructed from another stereo pair. Indeed it finds a comparable, even if consistently smaller, number of inliers with respect to other methods in poses from 1 to 4, while it finds drastically smaller inliers set in subsequent poses. This is mostly due to a worse response to calibration inaccuracies with respect to methods based on reprojection error.

The hybrid method has comparable performance to the coupled 3D2D method, and it finds a good number of inliers by employing reprojection error instead of object space error in the outliers rejection scheme, even if using a 3D3D approach in the computation of candidates solutions. However, like

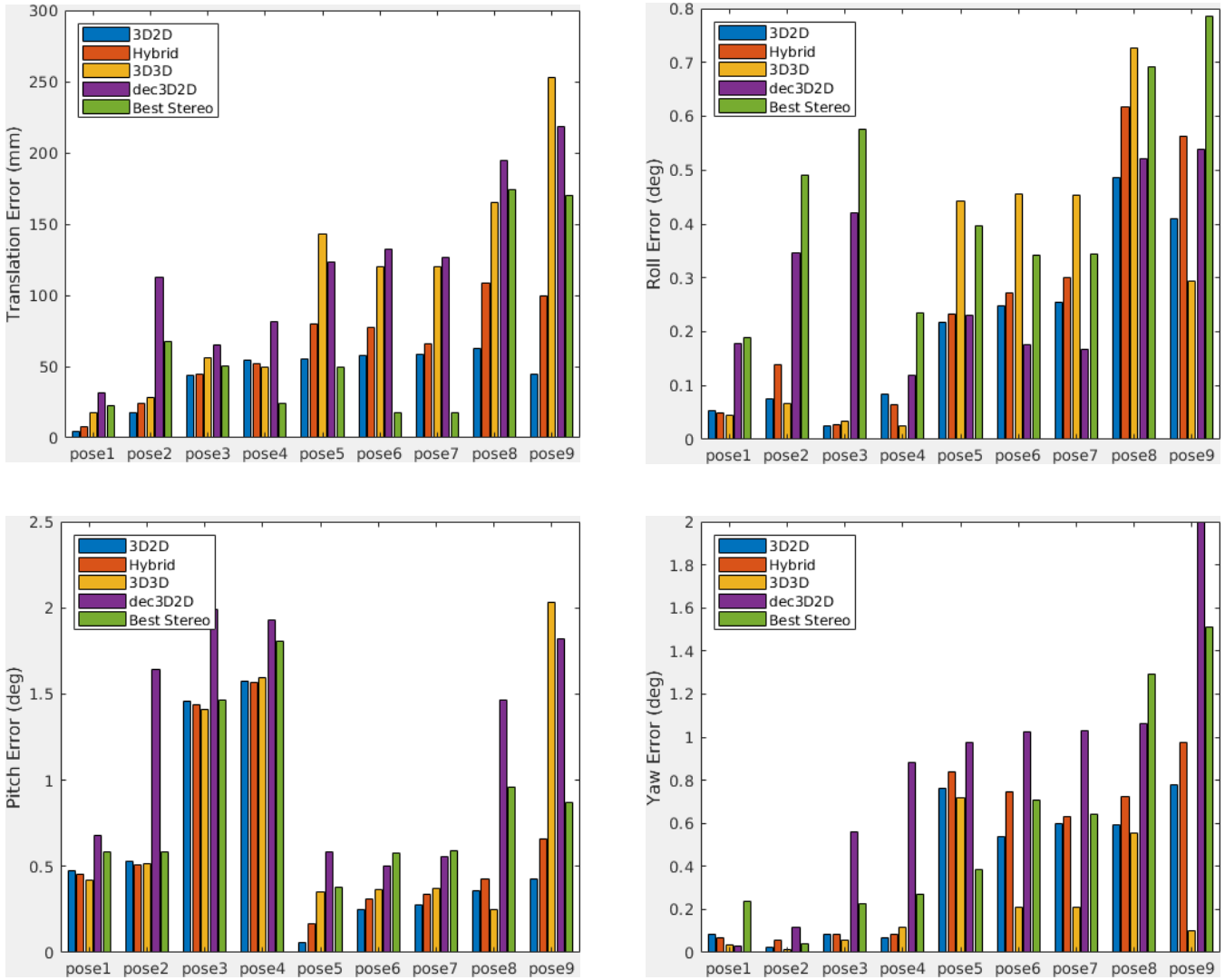


Figure 5.6: Localization errors in the relative pose estimation experiment.

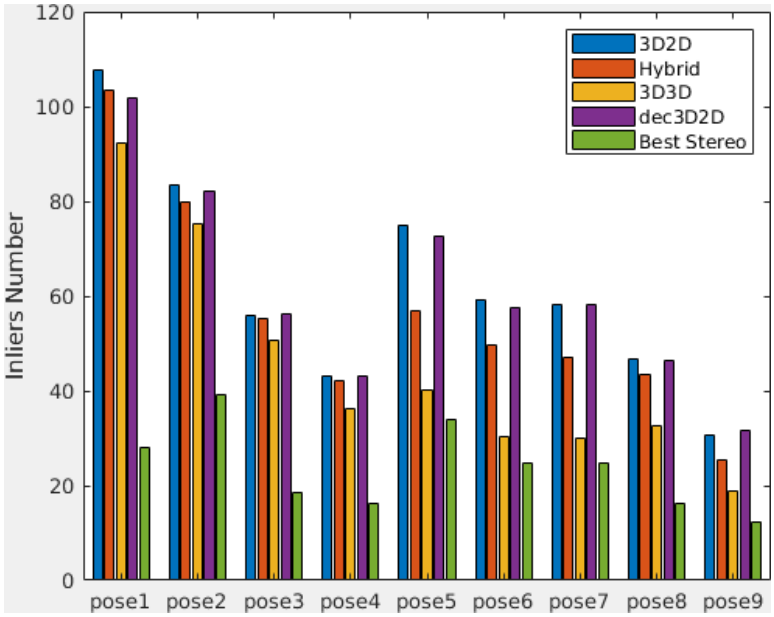


Figure 5.7: Inliers number in relative pose estimation.

the 3D3D method, it suffers more calibration inaccuracy as can be observed by the inlier number in rotated poses.

Finally let us consider the outliers rejection step of the decoupled 3D2D method, that consists in applying p3p inside one RANSAC instance for each camera. Since in our examples each camera observes a set of scene points reconstructed using data from a single stereo camera, this is reasonably one of the best methods in terms of number of found inliers. Indeed our reconstruction method is equivalent to separately perform reconstruction on each camera and then transform obtained point clouds according to calibration, in order to express them in the principal camera frame. The number of inliers found by the decoupled method is thus unaffected in this settings by bad extrinsics calibration. Interestingly the number of inliers found by the coupled 3D2D approach is always similar and sometimes slightly bigger than the number of inliers found using the decoupled method, while still using a reasonable reprojection error threshold in coupled outliers rejection.

Since the coupled 3D2D approach has shown to give the best localization performance, and highest number of inliers, both for small and wide change of pose, we conclude this is the best method in terms of accuracy among the ones run in this experiment.

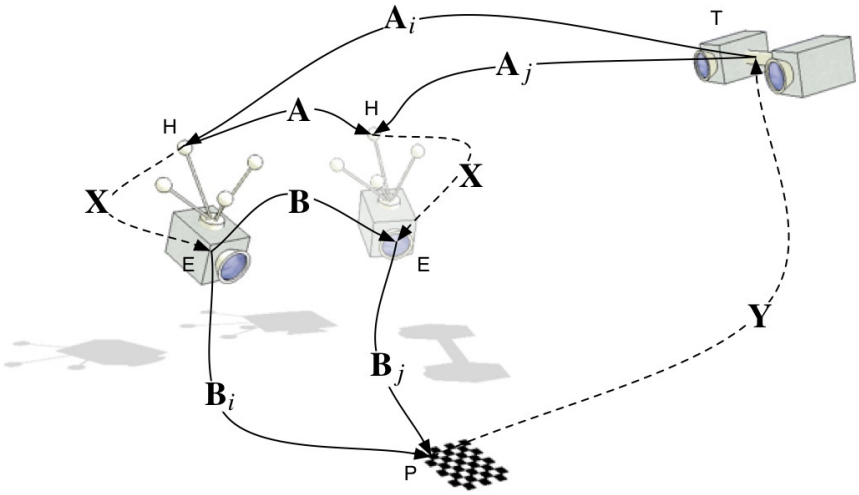


Figure 5.8: Hand eye calibration for marker to camera transform.

5.4 Visual Odometry

We run our visual odometry procedure on data acquired over a small path traveled in a room equipped with a motion capture system offering ground truth data. As in the previous section we start explaining how ground truth is obtained, then we give details about the experiment and the results.

5.4.1 Ground Truth

A visual marker is placed on our multi-stereo camera, and tracked by an external motion capture system. This gives the trajectory of a frame m placed on the marker, in the reference system o of the motion capture system. This ground truth is not directly applicable for comparison with the visual odometry agorithm, since the latter computes the principal camera frame trajectory in a reference system corresponding to the initial pose of the principal camera. Thus we need to obtain the ground truth camera trajectory from the maker trajectory computed by the motion capture system, and then express the estimated and ground truth trajectories in a common reference frame for comparison. We derive the transformation relating m to the principal camera frame by performing hand eye calibration using the tool of [59]. As shown in Figure 5.8, this consists in tracking the marker with the external motion capture system while observing a checkerboard pattern, giving for each detection a marker motion $\mathbf{A}$ and a camera motion $\mathbf{B}$ that

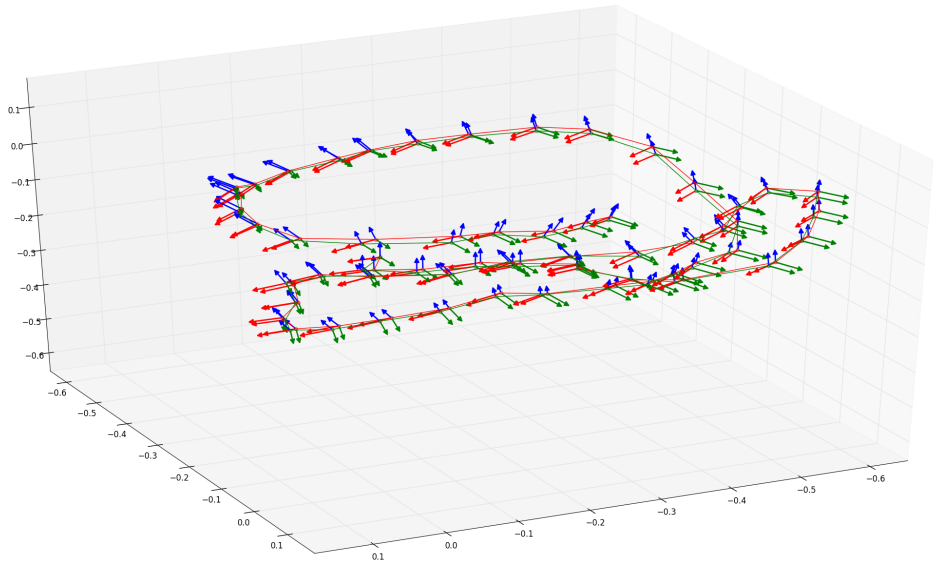


Figure 5.9: Aligned frames over the dataset employed in hand eye calibration. The marker transformed positions are represented by triads joined by a red line, while camera positions estimated with checkerboard pattern are represented by triads joined by a green line. Axis scale is in meters.

can be employed in forming a set of rigid body constraints of the form:

$$\mathbf{AX} = \mathbf{XB}. \tag{5.1}$$

We run the tool over a set of 590 frames video, detecting the checkerboard and localizing the camera with respect to it at each frame with a global average reprojection error of 0.2242 pixels and a standard deviation of 0.171 pixels. The tool then performs outliers rejection over camera and marker motion pairs related by rigid body constraint, resulting in 580 inliers. Position errors after alignment are shown in Figure 5.10, with an average translation error of 0.0106 meters and an average orientation error of 2.1147 degrees among pairs of camera and marker poses after alignment. We show the set of aligned poses in Figure 5.9.

For comparing estimation and ground truth, we first apply the hand eye transformation to the marker trajectory, and finally we perform alignment between the two trajectories. To this extent we employ the tool of [60], that solves absolute orientation between estimated and ground truth translations:

$$\min_{\mathbf{R},\mathbf{t}} \sum_{t} \|\mathbf{R}\mathbf{p}_{e,t} + \mathbf{t} - \mathbf{p}_{g,t}\|^2 \tag{5.2}$$

where $\mathbf{R}$, $\mathbf{t}$ represent the aligning Euclidean transformation, $\{\mathbf{p}_{g,t}\}_{t=1..T}$ is

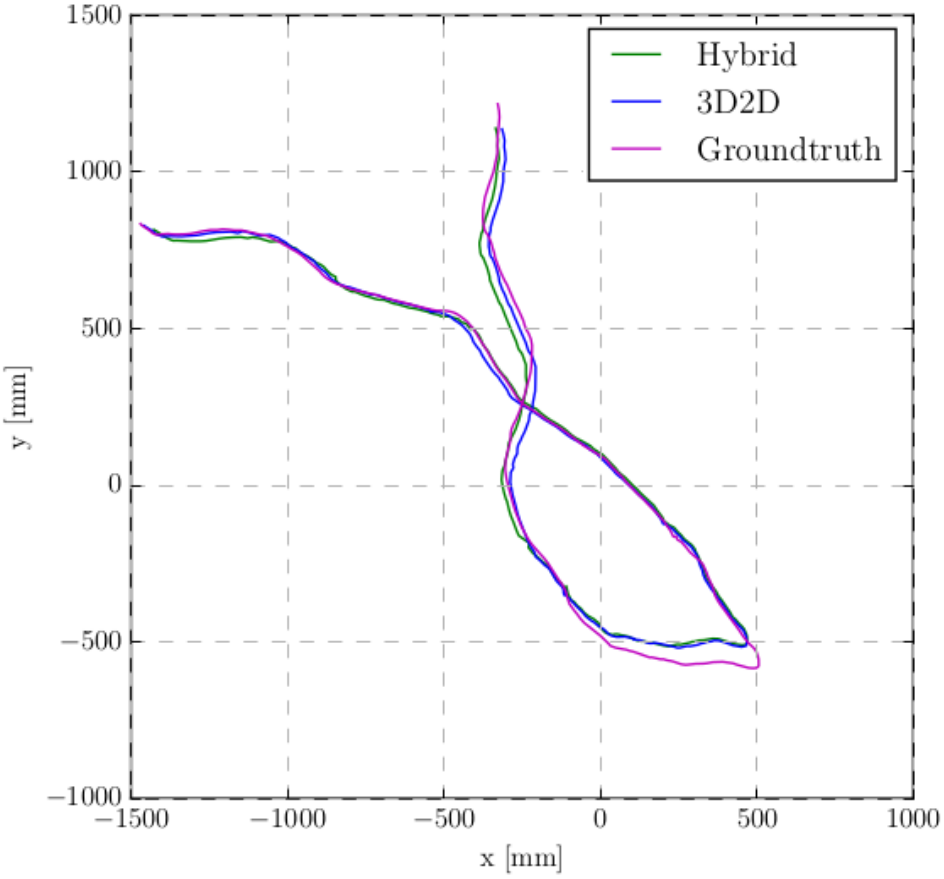
Figure 5.10: Translation and rotation alignment error over the dataset used to compute the transformation relating camera and marker frames.

the set of camera ground truth translations, and $\{\mathbf{p}_{e,t}\}_{t=1..T}$ is the set of camera estimated translations.

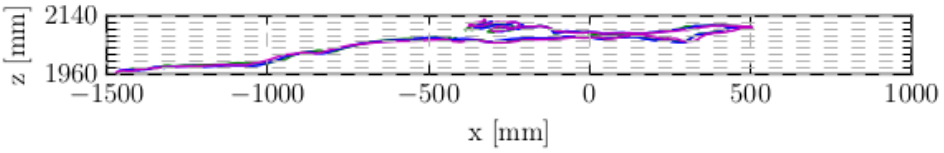
5.4.2 Experiment and Results

On each acquisition during the path tracked by the motion capture system we extract a uniformly distributed set of 70 SURF features on left images, while we do not constrain number and positioning of right images features. Then we apply stereo matching for scene reconstruction. Matching among consecutive frames is performed by projection of previous frame reconstruction under constant motion model, with a patch size of 20 pixels. We compare two different methods in the outliers rejection scheme, both based on reprojection error. The first one is the coupled 3D2D solution based on gP3P, while the second is the hybrid approach explained in Section 4.4.3, computing at each step a candidate aligning Euclidean transformations and keeping as inliers all elements with reprojection error below a fixed threshold. We set a threshold of 2 pixels for both methods. We perform at each step motion only windowed bundle adjustment with a window size of 6 frames. Estimated and ground truth trajectories over a path of about 5 meters are shown in Figure 5.11. The procedure can be improved due to its simplicity, indeed it never performs map to frame matching and does not optimize map points, however estimate and ground truth trajectories are similar enough, in particular in terms the side view. Also notice that the hand eye calibration error is comparable to the odometry error both in terms of translation and rotation, thus our ground truth might be not enough precise for such a comparison.

This experiment gives however the interesting insight that a multi-stereo sensor can be used in fully coupled fashion without using custom techniques solving the perspective-n-points problem over multiple projection centers, as the 3D2D approach does. Indeed we see by inspecting Figure 5.12 and Figure 5.13 that the hybrid approach has very similar performance, while considering point clouds alignment and reprojection error in the outliers rejection scheme. This is mostly motivated by the fact that outliers rejection is always performed between perceptions at consecutive frames, thus the better behavior of the coupled 3D2D method in the situation of wide movements we experimentally shown in previous section is not evident here. However, in a more complex localization pipeline comprising map to frame matches and loop closures, the difference between the two outliers rejection approaches might become evident.



(a) Trajectories seen from top.



(b) Trajectories seen from side.

Figure 5.11: Estimated and ground truth principal camera trajectories expressed in the motion capture system reference frame.

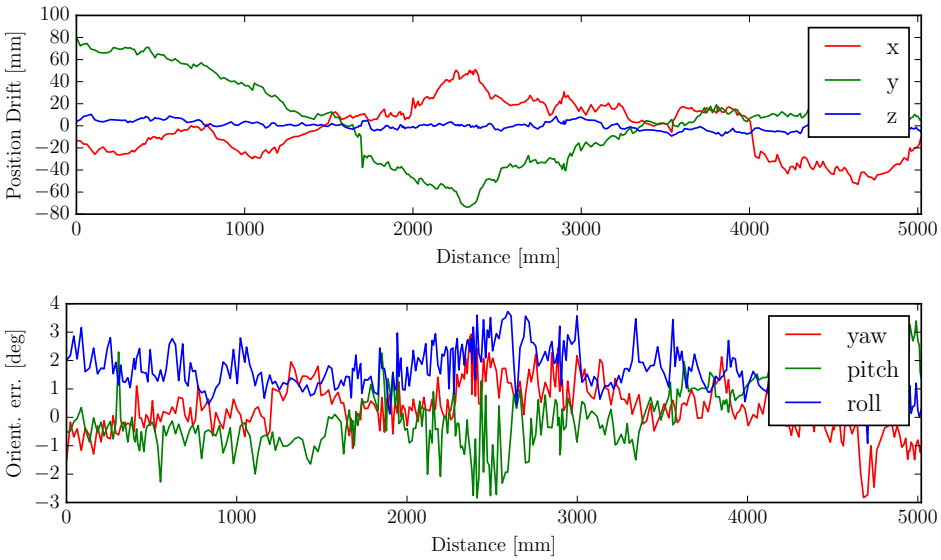


Figure 5.12: Visual odometry error of the coupled 3D2D relative pose estimation method.

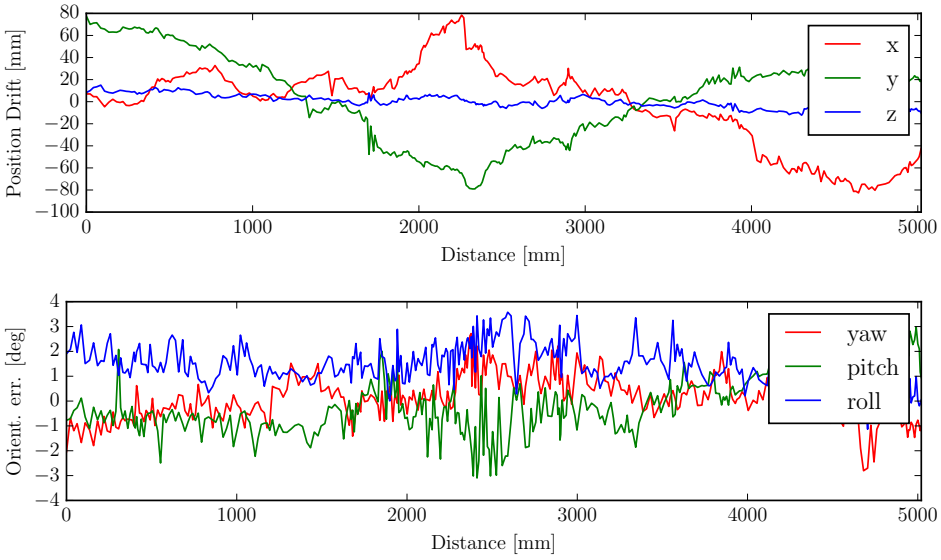


Figure 5.13: Visual odometry error of the hybrid relative pose estimation method.

Chapter 6

Conclusions and Future Works

We developed in this thesis the basic building blocks of a localization procedure for a system of multiple stereo cameras with non overlapping FOVs, consisting in reconstruction, matching and relative pose estimation from a scene model in terms of sparse point cloud associated with visual descriptors. Different relative pose estimation algorithms are compared on a real sensor composed of 4 stereo cameras, with the coupled 3D2D approach as the best performing.

The sensor has been calibrated using our extension to multi-stereo settings of a state of the art multi-camera calibration procedure, exploiting stereo information to increase the number of constraints on the searched solution, and FAIR-SURF to deal with high perspective effects on input images.

Finally, a minimal visual odometry procedure based on our algorithms has been developed, and compared with ground truth data obtained in laboratory. In these settings the hybrid relative pose estimation approach has shown to have comparable performance to the coupled 3D2D approach. Indeed the two algorithms only differs in the way in which outliers correspondences are rejected, where the coupled 3D2D approach employs the non-central projection model, and the hybrid exploits the stereo configuration to align point clouds obtained from each frame.

The next step is to insert the developed algorithms in a more complex localization pipeline. From the generality of the work, this might be either a VSLAM or a VO procedure, or the adaptation of an existing state of the art solution for stereo cameras to multi stereo settings. Common improve-

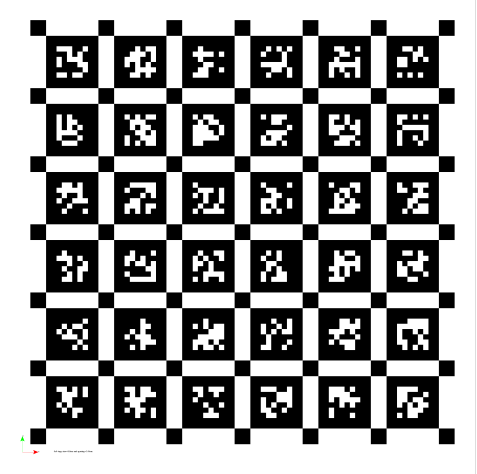


Figure 6.1: A checkerboard pattern composed of April tags [62]. This pattern is, differently from the classical checkerboard pattern, unambiguous and resistant to occlusion.

ments adopted in stereo settings like disabling translational contribution of points with high depth, or considering reconstructed points confidence can be included with no particular modifications too. As a further evolution, the developed algorithms can be employed on multiple RGB-D sensors stacked together, since these perform scene reconstruction from a single scene view as stereo cameras do.

Finally, regarding calibration, the proposed solution can be improved both in terms of accuracy and generality. We think it is possible to increase calibration accuracy by keeping the same pipeline and using a different calibration pattern. The calibration pattern should combine the unambiguity and occlusion resistance of the SURF pattern, together with detection accuracy of the checkerboard. These two qualities are combined in the patterns presented in [61–63]. These patterns are compared in [64], where tags are composed to form an occlusion resistant and unambiguous checkerboard. An example of such a pattern is shown in Figure 6.1.

This approach however impose constraints on relative positioning of different stereo sensors composing the rig. To increase generality a self-calibration procedure can be employed, and the multi-stereo configuration is well suited for this approach. Indeed data coming from each camera can be independently used to form environment maps without the need of an initialization phase for scale estimation. By obtaining correspondences between different maps, like for instance is done in [53], relative cameras pose can be obtained by maps alignment and refined in joint BA fashion.

Bibliography

[1] Zhengyou Zhang. *A flexible new technique for camera calibration*. 1998.

[2] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer Vision in C++ with the OpenCV Library*. O'Reilly Media, Inc., 2nd edition, 2013.

[3] Andrea Fusiello. Tutorial on rectification of stereo images. 11 1999.

[4] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.

[5] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[6] Edward Rosten, Gerhard Reitmayr, and Tom Drummond. Real-time video annotations for augmented reality. pages 294–302, 12 2005.

[7] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 778–792, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[8] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov 2004.

[9] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[10] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. pages 2564–2571, 11 2011.

[11] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. communications of the acm, 18(9), 509-517. *Commun. ACM*, 18:509–517, 09 1975.

[12] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *In VISAPP International Conference on Computer Vision Theory and Applications*, pages 331–340, 2009.

[13] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 518–529, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[14] Mirco Colosi Dominik Schlegel and Giorgio Grisetti. 2017.

[15] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[16] P. H. S. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78:138–156, 2000.

[17] Richard I. Hartley. Computation of the quadrifocal tensor. In *ECCV*, 1998.

[18] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A*, 4(4):629–642, Apr 1987.

[19] K.S. Arun, T.S. Huang, and Steven Blostein. Least-squares fitting of two 3-d point sets. ieee t pattern anal. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-9:698 – 700, 10 1987.

[20] Berthold K. P. Horn, Hugh M. Hilden, and Shahriar Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *J. Opt. Soc. Am. A*, 5(7):1127–1135, Jul 1988.

[21] Laurent Kneip, Davide Scaramuzza, and Roland Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. *CVPR 2011*, pages 2969–2976, 2011.

[22] C. . Lu, G. D. Hager, and E. Mjolsness. Fast and globally convergent pose estimation from video images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):610–622, June 2000.

[23] J. A. Hesch and S. I. Roumeliotis. A direct least-squares (dls) method for pnp. In *2011 International Conference on Computer Vision*, pages 383–390, Nov 2011.

[24] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81:155–166, 2009.

[25] Yinqiang Zheng, Yubin Kuang, Shigeki Sugimoto, Karl Åström, and Masatoshi Okutomi. Revisiting the pnp problem: A fast, general and optimal solution. pages 2344–2351. Computer Vision Foundation, 2013. The authoritative version of this paper will be available i IEEE Xplore.

[26] Steffen Urban, Jens Leitloff, and Stefan Hinz. Mlpnp - a real-time maximum likelihood solution to the perspective-n-point problem. In *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, volume 3, pages 131–138, 2016.

[27] Jose-Luis Blanco. A tutorial on SE(3) transformation parameterizations and on-manifold optimization. Technical Report 012010, University of Malaga, 2010.

[28] R. Mur-Artal and J. D. Tardos. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, Oct 2017.

[29] Dominik Schlegel and Giorgio Grisetti. HBST: A hamming distance embedding binary search tree for feature-based visual place recognition. *IEEE Robotics and Automation Letters*, 3(4):3741–3748, 2018.

[30] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3607–3613, Shanghai, China, May 2011.

[31] Raul Mur-Artal and Juan D. Tardós. Fast relocalisation and loop closing in keyframe-based SLAM. In *ICRA*, pages 846–853. IEEE, 2014.

[32] Dorian Galvez-Lopez and J. D. Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, October 2012.

[33] T. Kazik, L. Kneip, J. Nikolic, M. Pollefeys, and R. Siegwart. Real-time 6d stereo visual odometry with non-overlapping fields of view. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1529–1536, June 2012.

[34] M. E. Ragab and K. H. Wong. Multiple nonoverlapping camera pose estimation. In *2010 IEEE International Conference on Image Processing*, pages 3253–3256, Sep. 2010.

[35] G. Zhou, J. Ye, W. Ren, T. Wang, and Z. Li. On-board inertial-assisted visual odometer on an embedded system. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2602–2608, May 2014.

[36] G. Zhou, A. Liu, K. Yang, T. Wang, and Z. Li. An embedded solution to visual mapping for consumer drones. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 670–675, June 2014.

[37] Michael J. Tribou, Adam Harmat, David W.L. Wang, Inna Sharf, and Steven L. Waslander. Multi-camera parallel tracking and mapping with non-overlapping fields of view. *The International Journal of Robotics Research*, 34(12):1480–1500, 2015.

[38] Adam Harmat, Michael Trentini, and Inna Sharf. Multi-camera tracking and mapping for unmanned aerial vehicles in unstructured environments. *Journal of Intelligent and Robotic Systems*, 78:291–317, 2015.

[39] Adam Harmat, Inna Sharf, and Michael Trentini. Parallel tracking and mapping with multiple cameras on an unmanned aerial vehicle. In Chun-Yi Su, Subhash Rakheja, and Honghai Liu, editors, *Intelligent Robotics and Applications*, pages 421–432, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[40] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[41] Shaowu Yang, Sebastian A. Scherer, Xiaodong Yi, and Andreas Zell. Multi-camera visual slam for autonomous navigation of micro aerial vehicles. *Robotics and Autonomous Systems*, 93:116 – 134, 2017.

[42] S. Houben, J. Quenzel, N. Krombach, and S. Behnke. Efficient multi-camera visual-inertial slam for micro aerial vehicles. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1616–1622, Oct 2016.

[43] M. D. Grossberg and S. K. Nayar. A general imaging model and a method for finding its parameters. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 108–115 vol.2, July 2001.

[44] R. Pless. Using many cameras as one. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–587, June 2003.

[45] Gim Hee Lee, Bo Li, Marc Pollefeys, and Friedrich Fraundorfer. Minimal solutions for the multi-camera pose estimation problem. *The International Journal of Robotics Research*, 34(7):837–848, 2015.

[46] D. Nister. A minimal solution to the generalised 3-point pose problem. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I, June 2004.

[47] Chu-Song Chen and Wen-Yan Chang. On pose recovery for generalized visual sensors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(7):848–861, July 2004.

[48] Gerald Schweighofer and Axel Pinz. Globally optimal o(n) solution to the pnp problem for general camera models. In *BMVC*, 2008.

[49] Andreas Ess, Alexander Neubeck, and Luc Van Gool. Generalised linear pose estimation. In *BMVC*, 2007.

[50] L. Kneip, P. Furgale, and R. Siegwart. Using multi-camera systems in robotics: Efficient solutions to the npnp problem. In *2013 IEEE International Conference on Robotics and Automation*, pages 3770–3776, May 2013.

[51] Jun-Sik Kim, Myung Hwangbo, and Takeo Kanade. Motion estimation using multiple non-overlapping cameras for small unmanned aerial vehicles. In *2008 IEEE International Conference on Robotics and Automation*, pages 3076–3081, May 2008.

[52] Sandro Esquivel, Felix Woelk, and Reinhard Koch. Calibration of a multi-camera rig from non-overlapping views. In Fred A. Hamprecht, Christoph Schnörr, and Bernd Jähne, editors, *Pattern Recognition*, pages 82–91, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[53] G. Carrera, A. Angeli, and A. J. Davison. Slam-based automatic extrinsic calibration of a multi-camera rig. In *2011 IEEE International Conference on Robotics and Automation*, pages 2652–2659, May 2011.

[54] Renbo Xia, Maobang Hu, Jibin Zhao, Songlin Chen, and Yueling Chen. Global calibration of multi-cameras with non-overlapping fields of view based on photogrammetry and reconfigurable target. *Measurement Science and Technology*, 29(6):065005, apr 2018.

[55] T. Straub, J. Ziegler, and J. Beck. Calibrating multiple cameras with non-overlapping views using coded checkerboard targets. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 2623–2628, Oct 2014.

[56] B. Li, L. Heng, K. Koser, and M. Pollefeys. A multiple-camera system calibration toolbox using a feature descriptor-based calibration pattern. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1301–1307, Nov 2013.

[57] Richard Hartley, Jochen Trumpf, Yuchao Dai, and Hongdong Li. Rotation averaging. *International Journal of Computer Vision*, 103(3):267–305, Jul 2013.

[58] Yanwei Pang, Wei Li, Yuan Yuan, and Jing Pan. Fully affine invariant surf for image matching. *Neurocomput.*, 85:6–10, May 2012.

[59] Fadri Furrer, Marius Fehr, Tonci Novkovic, Hannes Sommer, Igor Gilitschenski, and Roland Siegwart. *Evaluation of Combined Time-Offset Estimation and Hand-Eye Calibration on Robotic Datasets*. Springer International Publishing, Cham, 2017.

[60] Zichao Zhang and Davide Scaramuzza. A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2018.

[61] M Fiala. Artag, a fiducial marker system using digital techniques. volume 2, pages 590 – 596 vol. 2, 07 2005.

[62] E. Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation*, pages 3400–3407, May 2011.

[63] Bradley Atcheson, Felix Heide, and Wolfgang Heidrich. Caltag: High precision fiducial markers for camera calibration. pages 41–48, 01 2010.

[64] Artur Sagitov, Ksenia Shabalina, Leysan Sabirova, Hongbing Li, and Evgeni Magid. Artag, apriltag and caltag fiducial marker systems: Comparison in a presence of partial marker occlusion and rotation. pages 182–191, 01 2017.

Appendix A

Position Representation

A rigid body position is represented as a 3 DoF translation plus a 3 DoF rotation with respect to some reference frame o. More precisely, this is done by placing a reference system b at some point of the rigid body, considered as rigid body origin. This point has coordinate $\mathbf{t}_b^o$ in the reference system o, and represents the rigid body translation. The orientation of b in o is the rigid body orientation, and different alternatives are available for its representations, each one presenting advantages and drawbacks depending on the application. Here we focus on three different ways of representing rotations: matrix form, quaternions, and Rodrigues representation.

A.1 Matrix form

A convenient way of representing a rigid body orientation is to write the b axis versors expressions in o. The coordinates of each versor are simply the cosines of the angles the versor forms with respect the corresponding o axis. Let these be $\mathbf{x}_b$, $\mathbf{y}_b$, $\mathbf{z}_b$, the rotation matrix representing the rotation of b in o is obtained by horizontally stacking them:

$$\mathbf{R}_b^o = \begin{bmatrix} \mathbf{x}_b & \mathbf{y}_b & \mathbf{z}_b \end{bmatrix}. \tag{A.1}$$

Note that the columns vectors of $\mathbf{R}_b^o$ are normal and mutually orthogonal, since they represents the unit versors of an orthonormal frame. Thus rotation matrices are orthonormal matrix.

Consider the situation in which $\mathbf{t}_b^o = 0$ (the origins of b and o coincide), and a point with coordinates $\mathbf{X}^o = (X_o\, Y_o\, Z_o)$ in o and $\mathbf{X}^b = (X_b\, Y_b\, Z_b)$ in b. The following relation holds:

$$\mathbf{X}^o = X_b\, \mathbf{x}_b + Y_b\, \mathbf{y}_b + Z_b\, \mathbf{z}_b = \mathbf{R}_b^o\, \mathbf{X}^b. \tag{A.2}$$

Thus the rotation matrix also represents the transformation that maps b coordinates to o coordinates. For the orthonormal property, the inverse relation can be simply obtained by transposing $\mathbf{R}_b^o$:

$$\mathbf{X}^b = \mathbf{R}_o^b \, \mathbf{X}^o = \mathbf{R}_b^{o\,T} \, \mathbf{X}^o. \tag{A.3}$$

In the general situation in which $\mathbf{t}_b^o$ may be different from the null vector, the coordinates transformation takes the form:

$$\mathbf{X}^o = \mathbf{R}_b^o \, \mathbf{X}^b + \mathbf{t}_b^o, \tag{A.4}$$

that can be written, by representing points with their homogeneous expression, as

$$\tilde{\mathbf{X}}^o = \mathbf{T}_b^o \, \tilde{\mathbf{X}}^b = \begin{bmatrix} \mathbf{R}_b^o & \mathbf{t}_b^o \\ \mathbf{0}^T & 1 \end{bmatrix} \tilde{\mathbf{X}}^b \tag{A.5}$$

where $\tilde{\mathbf{X}}^o = (X_o \, Y_o \, Z_o \, 1)^T$ and $\tilde{\mathbf{X}}^b = (X_b \, Y_b \, Z_b \, 1)^T$.

A.2 Quaternions

Quaternions may be thought as complex numbers with three different imaginary parts. A quaternion with real part q_0, and imaginary parts q_x, q_y, and q_z can be written as:

$$\check{q} = q_0 + iq_x + jq_y + kq_z. \tag{A.6}$$

The quaternions multiplication is obtained by the product of their expression, where the imaginary parts products follow the rules:

$$\begin{aligned} &i^2 = -1, && j^2 = -1, && k^2 = -1, \\ &ij = k, && jk = i, && ki = j, \\ &ji = -k, && kj = -i, && ik = -j. \end{aligned}$$

The conjugate of the quaternion $\check{q}$ is denoted as $\check{q}^*$, and is obtained by negation of the imaginary coefficients.

By thinking of quaternions as four dimensional vectors $(q_0, q_x, q_y, q_z)^T$ we obtain the quaternions dot product operation and quaternion norm. A quaternion is unitary if its norm equals 1.

Nothing that

$$\check{q}\check{q}^* = q_0^2 + q_x^2 + q_y^2 + q_z^2 = \check{q} \bullet \check{q}, \tag{A.7}$$

the inverse of a nonzero quaternion is:

$$\check{q}^{-1} = \frac{\check{q}^*}{\check{q} \bullet \check{q}} \tag{A.8}$$

that for a unitary quaternion is just the conjugate.

Given a point $\mathbf{X} = (X\,Y\,Z)^T$, we can represent it as a purely imaginary quaternion $\check{X} = iX + jY + kZ$, and transform it using the unitary quaternion $\check{q}$ via composite product, obtaining a purely imaginary quaternion:

$$\check{X}' = \check{q}\check{X}\check{q}^*. \tag{A.9}$$

This operation preserves magnitude, dot product and cross product. As consequence the composite product by a unitary quaternion represents a rotation.

Now it is useful to define the quaternion product in matrix form:

$$\begin{aligned} \check{q}\check{r} &= \begin{bmatrix} q_0 & -q_x & -q_y & -q_z \\ q_x & q_0 & -q_z & q_y \\ q_y & q_z & q_0 & -q_x \\ q_z & -q_y & q_x & q_0 \end{bmatrix} = \mathcal{Q}\check{r}, \\ \check{r}\check{q} &= \begin{bmatrix} q_0 & -q_x & -q_y & -q_z \\ q_x & q_0 & q_z & -q_y \\ q_y & -q_z & q_0 & q_x \\ q_z & q_y & -q_x & q_0 \end{bmatrix} = \bar{\mathcal{Q}}\check{r}. \end{aligned} \tag{A.10}$$

Notice that the matrices associated with the conjugate quaternion are simply obtained by transposition. By employing the matrix notation for quaternion multiplication in A.9 we obtain:

$$\check{X}' = \bar{\mathcal{Q}}^T\mathcal{Q}\check{X} \tag{A.11}$$

where the transformation matrix applied to $\check{X}$ has the form:

$$\bar{\mathcal{Q}}\mathcal{Q} = \begin{bmatrix} \check{q}\bullet\check{q} & 0 & 0 & 0 \\ 0 & (q_0^2 + q_x^2 - q_y^2 - q_z^2) & 2(q_xq_y - q_0q_z) & 2(q_xq_z + q_0q_y) \\ 0 & 2(q_yq_x + q_0q_z) & (q_0^2 - q_x^2 + q_y^2 - q_z^2) & 2(q_yq_z - q_0q_x) \\ 0 & 2(q_zq_x - q_0q_y) & 2(q_zq_y + q_0q_x) & (q_0^2 - q_x^2 - q_y^2 + q_z^2) \end{bmatrix}.$$

The rotation matrix $\mathbf{R}$ satisfying $\mathbf{X}' = \mathbf{R}\mathbf{X}$ is the bottom-right 3×3 component of $\bar{\mathcal{Q}}\mathcal{Q}$. Since $\check{q}$ is normal, its corresponding matrices are orthonormal and $\mathbf{R}$ is orthonormal.

A.3 Rodrigues Representation

The main drawback of rotation matrices and quaternions is that they are not minimal, meaning that they use a number of parameters greater than

the DoF of a rigid body orientation in the space. Indeed to be well defined a rotation matrix needs to be orthonormal (three orthogonality constraints and three unitary norm constraints), and a quaternion needs to be unitary (one constraint), leaving in both situations three DoF.

Rodrigues representation is instead minimal, representing a rotation as a three-dimensional vector $\mathbf{r}$. The $\mathbf{r}$ direction represents the rotation axis, while its magnitude the counter clockwise rotation around it. For obtaining a relation between rotation matrices and Rodrigues representation, it is better to first explain another non minimal representation called axis-angle. This representation uses four parameters: a three dimensional unitary vector $\mathbf{a}$ for the rotation axis, and a fourth parameter θ for the rotation angle. The relation between axis-angle and Rodrigues representations is straightforward:

$$\begin{aligned} \theta &= \|\mathbf{r}\|, \\ \mathbf{a} &= \frac{\mathbf{r}}{\|\mathbf{r}\|}. \end{aligned} \tag{A.12}$$

We now derive, given a axis $\mathbf{a}$ and an angle θ, the corresponding rotation matrix $\mathbf{R}(\mathbf{a}, \theta)$. This can be done by composing three rotations:

- $\mathbf{C}$, that aligns the $\mathbf{z}$ reference frame axis to $\mathbf{a}$
- A counter clockwise rotation of θ around the $\mathbf{z}$ axis
- $\mathbf{C}^T$, the inverse rotation of $\mathbf{C}$

The matrix $\mathbf{C}$ maps the $(0\,0\,1)^T$ versor to $\mathbf{a}$, thus it can be written in the following way:

$$\mathbf{C} = \begin{bmatrix} \mathbf{n} & \mathbf{s} & \mathbf{a} \end{bmatrix} \tag{A.13}$$

for some pair of orthogonal unitary vectors $\mathbf{n}$ and $\mathbf{s}$ satisfying $\mathbf{n} \times \mathbf{s} = \mathbf{a}$. The rotations composition has the form:

$$\begin{aligned} \mathbf{R}(\mathbf{a}, \theta) &= \begin{bmatrix} \mathbf{n} & \mathbf{s} & \mathbf{a} \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{n}^t \\ \mathbf{s}^T \\ \mathbf{a}^T \end{bmatrix} \\ &= \mathbf{a}\mathbf{a}^t + \left(\mathbf{n}\mathbf{n}^T + \mathbf{s}\mathbf{s}^T\right)\cos\theta + \left(\mathbf{s}\mathbf{n}^T + \mathbf{n}\mathbf{s}^T\right)\sin\theta. \end{aligned} \tag{A.14}$$

The above expression can be written as a function of the only parameters $\mathbf{a}$ and θ by nothing that:

$$\begin{aligned} \mathbf{n}\mathbf{n}^T + \mathbf{s}\mathbf{s}^T &= \mathbf{I} - \mathbf{r}\mathbf{r}^T, \\ \mathbf{s}\mathbf{n}^T + \mathbf{n}\mathbf{s}^T &= [\mathbf{a}]_\times, \end{aligned} \tag{A.15}$$

where $\mathbf{I}$ is the 3×3 identity matrix, and $[\mathbf{a}]_\times$ is the skew symmetric matrix representing the cross product operator related to $\mathbf{a}$. In the end the rotation matrix related to an axis-angle pair takes the form:

$$\mathbf{R}(\mathbf{a}, \theta) = \mathbf{a}\mathbf{a}^T + (\mathbf{I} - \mathbf{a}\mathbf{a}^T)\cos\theta + [\mathbf{a}]_\times \sin\theta. \tag{A.16}$$

By exploiting equation A.12, and the relation $\mathbf{r}\mathbf{r}^T = \mathbf{I}\|\mathbf{r}\|^2 + [\mathbf{r}]_\times^2$, the rotation matrix $\mathbf{R}(\mathbf{r})$ equivalent to the Rodrigues orientation $\mathbf{r}$ has expression:

$$\mathbf{R}(\mathbf{r}) = \mathbf{I} + \frac{\sin\theta}{\theta}[\mathbf{r}]_\times + \frac{1-\cos\theta}{\theta^2}[\mathbf{r}]_\times^2 . \tag{A.17}$$

In practical implementations this should be approximated with $\mathbf{I} + [\delta\mathbf{r}]_\times$ when the $\mathbf{r}$ magnitude is small, where $\delta\mathbf{r}$ is a small perturbation of $\mathbf{r}$.

The inverse relation has the following form:

$$\begin{aligned} \theta &= \cos^{-1}\left(\frac{Tr(\mathbf{R}) - 1}{2}\right), \\ \mathbf{r} &= \frac{\theta}{2\sin\theta}\begin{bmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix}, \end{aligned} \tag{A.18}$$

where R_{ij} is the element at i-th row and j-th column of $\mathbf{R}$. Note that this equation is only defined for $\theta \neq 0 + k\pi$ for each $k \in \mathbb{N}$.

Appendix B

DJI Guidance Visual Odometry

The DJI Guidance multi-stereo camera comes out of the box with a visual-inertial odometry procedure running on a computer mounted on the sensor. As explained in [35, 36], this works in 3D2D fashion in the following way. FAST features associated with BRIEF descriptors are computed from each image, and stereo matching is performed followed by triangulation. Based on the movement guess given by the IMU, 3D2D matches are obtained over subsequent views and, from this, the translation factor is refined by assuming the rotation computed by the IMU to be correct. As a result the PnP problem becomes linear and has closed form solution. The movements computed by each camera in decoupled fashion are then joined by complementary filtering.

We aim to compare trajectories estimate obtained by our VO procedure with the ones computed by the algorithm running on the Guidance sensor. We also want to compare the two estimate with ground truth data given by a motion capture system.

However each of the three trajectory computation methods has its own reference system:

i. Our algorithm computes the trajectory of the principal camera frame.

ii. The Guidance algorithm computes the trajectory of a different frame, that we call robot frame.

iii. The motion capture system computes the trajectory of the visual marker placed on the sensor.

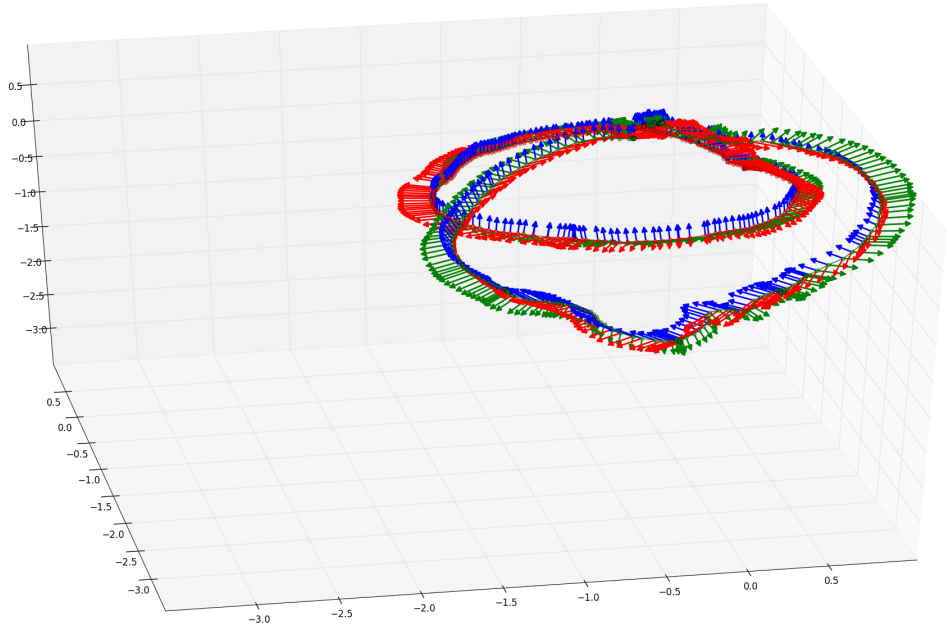


Figure B.1: Result of the hand eye calibration procedure. Trajectories estimated by the motion capture system and the Guidance visual odometry procedure are fed to the tool of [59]. Clearly the two trajectories are very different and the tool does not find any solution. Axis scales are in meters.

For comparison, we need to transform estimated trajectories in such a way they all represent the movement of the same frame, rigidly attached to the sensor. We choose this frame to be the one of the visual marker mounted on the sensor and tracked by the motion capture system.

In Section 5.4 the transformation relating the principal camera center to the visual marker placed on the Guidance sensor has been obtained. Thus we apply hand eye calibration over a small trajectory to compute the transformation that relates the robot frame to the marker frame. While moving the sensor we store trajectories computed by the motion capture system and the Guidance sensor, then we fed them to the hand eye calibration tool of [59]. Aligned trajectories are shown in Figure B.1, while trajectory alignment error is shown in Figure B.2. Clearly the trajectory estimate obtained by the Guidance VO procedure suffers of high drift, thus no hand eye solution can be obtained in this settings.

We think this is mostly due to our sensor setup. Our multi-stereo camera is mounted on a custom bracket, and attached to another block containing the IMU. Thus relative positioning between IMU and cameras is not the same as the one developed by producer, and default calibration data does

not represent our system configuration. Since the Guidance software does not provide any interface for calibrating the sensor, and it is unclear if it performs internally self-calibration (e.g.in hand-eye fashion), we think the high drift is due to wrong calibration parameters.

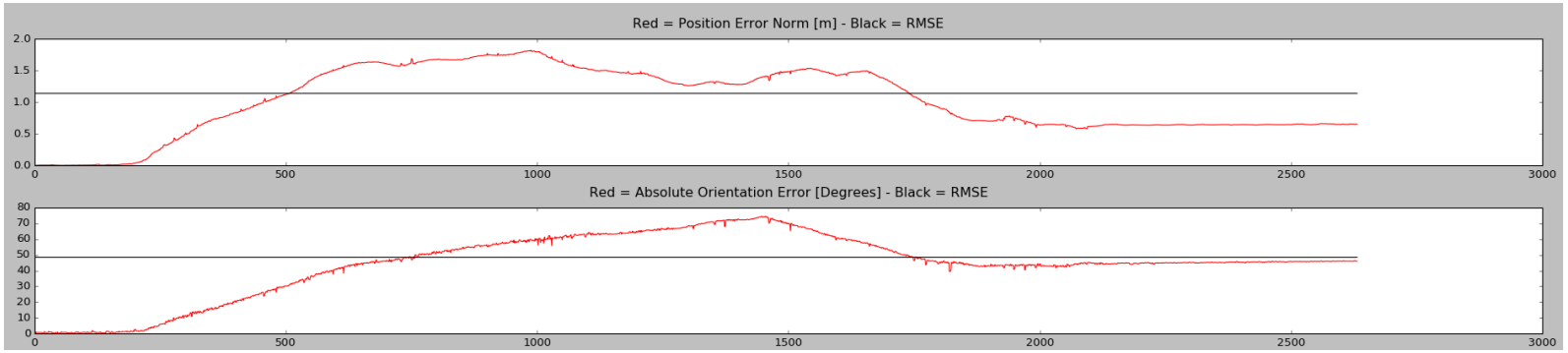


Figure B.2: Alignment error over trajectories computed by the motion capture system and Guidance VO solution.