Politecnico di Torino

Master of Science Degree in MECHATRONIC ENGINEERING

Master Thesis

Comparison of Stereo Visual Inertial Odometry Algorithms for Unmanned Ground Vehicles

Supervisor:
prof. Marcello Chiaberge

Candidate:
Roberto Cappellaro

26 July 2019

Abstract

PIC4SeR is the Interdepartmental center of PoliTO for service robotics. For its research it emerged the need to investigate indoor localization algorithms, in particular the visual-inertial type. This work aims to study different types of algorithms to assess which one is the best choice for indoor localization with the already available COTS hardware.
Although the end application is meant to be UAV, a Jackal UGV is used instead, because it was the vehicle available and it lowered the risks of damages during the testing phase. A MYNTYEYE S stereo camera, with included IMU and IR projector, was available and mounted on the UGV.
Three algorithms are considered: a light-weight filter-based VIO framework, ROVIO, and two optimization-based VIO frameworks, VINS-Fusion and OKVIS, that should better accommodate data asynchrony.
The algorithms are tested in two different environments making the robot follow two paths multiple times: a linear path in a corridor and a pseudo-rectangular one in a room.
The algorithms performance is evaluated by two parameters: the relative error on the total traveled distance and the difference between initial and final position of the UGV.
The tests underlined no best algorithm, but a dependence on the environment.
As future work, it would be interesting to test a camera using a tight hardware-synchronization with an IMU, since, according to the literature, should be a big source of error.

Contents

1 Introduction **1**
1.1 Objectives . . . 1

2 Stereo camera **2**
2.1 MYNTEYE S1030-IR-120/Mono . . . 2
2.2 Camera calibration . . . 2
2.2.1 Extrinsic and intrinsic camera model . . . 4
2.2.2 Pinhole projection model . . . 5
2.2.3 Lens distortion models . . . 6
Radial-tangential distortion model . . . 6
Kannala-Bradt distortion model . . . 7
2.2.4 Best camera calibration practices . . . 8
2.3 Camera calibration with Kalibr . . . 9
2.3.1 Camera calibration procedure used . . . 10
2.3.2 Kalibr setting and result analysis . . . 11
2.4 Stereo triangulation . . . 11
2.4.1 Epipolar geometry . . . 11
2.4.2 Stereo triangulation . . . 13
2.4.3 Disparity space . . . 14

3 Elements of visual odometry **15**
3.1 VO approaches . . . 15
3.1.1 Feature based approach . . . 15
3.1.2 Appearance based approach . . . 16
3.1.3 Image pyramids . . . 17
3.2 FAST: Features from Accelerated Segment Test . . . 18
3.3 Harris corner detector . . . 19
3.3.1 Harris measure . . . 20
3.3.2 Shi-Tomasi measure . . . 20
3.4 KLT feature tracker . . . 21
3.4.1 Standard implementation . . . 22
3.4.2 Iterative implementation . . . 23
3.5 BRISK . . . 24
3.5.1 Keypoint descriptor . . . 24

Sampling pattern and rotation estimation . . . 24
Descriptor building . . . 25
3.5.2 Descriptor matching . . . 25

4 Stereo visual inertial odometry algorithms 26
4.1 Notation . . . 26
4.2 ROVIO . . . 27
4.2.1 Representation of 3D rotations . . . 27
4.2.2 Representation of 3D unit vectors . . . 29
4.2.3 Robocentric state representation . . . 31
4.2.4 Projection model and linear warping . . . 32
4.2.5 Photometric error and patch alignment . . . 32
4.2.6 Detection and scoring . . . 33
4.2.7 Iterated extended Kalman filtering . . . 34
4.2.8 Filter setup and state definition . . . 35
4.2.9 State propagation . . . 36
4.2.10 Direct innovation term and update . . . 38
4.2.11 Landmark tracking . . . 39
4.2.12 Landmark management . . . 40
4.2.13 Stereo camera setup . . . 40
4.3 VINS-Fusion . . . 41
4.3.1 Local and global sensors . . . 42
4.3.2 Stereo VIO module . . . 42
Vision pre-processing . . . 42
IMU pre-integration . . . 43
Visual inertial estimator initialization . . . 46
Gyroscope bias calibration . . . 46
States . . . 47
Cost function . . . 47
Camera factor . . . 48
IMU factor . . . 48
Other factors . . . 48
Optimization . . . 48
Marginalization . . . 49
Failure detection module . . . 50
4.4 OKVIS . . . 51
4.4.1 States . . . 51
4.4.2 Non-linear optimization . . . 52
4.4.3 Reprojection error . . . 52
4.4.4 IMU kinematics . . . 52
4.4.5 IMU measurement error term . . . 53
4.4.6 Keypoint matching and keyframe selection . . . 54
Partial marginalization . . . 54

5 Tests and results analysis **57**
5.1 Tests 57
5.2 Results analysis 59

Bibliography **64**

List of Figures

2.1 MYNTEYE S camera. 4
2.2 Camera projection model. 4
2.3 Pinhole projection model. 5
2.4 Radial-tangential distortions. 7
2.5 Fisheye camera model. 8
2.6 Calibration target. 10
2.7 Confront between reprojection errors. 12
2.8 Epipolar geometry model. 13
2.9 Stereo triangulation. 13

3.1 Representation of an image pyramid. 18
3.2 FAST algorithm implementation. 19
3.3 Harris and Shi-Tomasi measures. 21
3.4 BRISK sampling pattern. 25

4.1 Representation of 3D unit vectors. 31
4.2 World centered vs camera centered state representation. 32
4.3 ROVIO framework diagram. 37
4.4 ROVIO landmark tracking. 39
4.5 VINS-Fusion framework visual representation. 42
4.6 VINS-Fusion marginalization strategy. 50
4.7 OKVIS: marginalization of the first frames. 55
4.8 OKVIS: a regular frame is marginalized out. 55
4.9 OKVIS: the oldest keyframe is marginalized out. 56

5.1 Jackal UGV with MYNTEYE camera. 58
5.2 Trajectory comparison 200 Hz - IR off, first trial. 60
5.3 Trajectory comparison of the best settings for each algorithm in the corridor environment. 63
5.4 Trajectory comparison of the best settings for each algorithm in the room environment. 63

List of Tables

2.1 MYNTEYE S camera-IMU asynchrony. 3
2.2 MYNTEYE S1030-IR datasheet. 3

5.1 Room results. 60
5.2 Corridor results. 61
5.3 Corridor VIO comparison. 61
5.4 Room VIO comparison. 61
5.5 Room average results. 62
5.6 Corridor average results. 62

Chapter 1

Introduction

PIC4Ser is the PoliTO Interdepartmental Centre for Service Robotics. Its objective it to connect and coordinate the activity of researchers with different engineering background, in order to develop new technologies in the complex service robotic field. The research covers different areas and one consists in the development of technologies and programs for indoor localization of UAV and UGV. For this task it is necessary to have a robust and accurate indoor localization and this is where this thesis project came to be. In fact this works aims to study different types of visual inertial algorithms to assess which one is the best choice for indoor localization with the already available Commercial Off-the-Shelf (COTS) hardware. Despite the fact that the end application is intended for aerial vehicles, a Jackal Unmanned Ground Vehicle (UGV) was used for the tests, since it was the vehicle available and it lowered the risks of damages during the maneuvers. The sensor used was a MYNTYEYE S stereo camera, that comes already built with a inertial measurement unit inside and two infrared lights to project a pattern of points for each monocular camera. One of the stereo visual inertial framework considered is ROVIO, a light-weight filter-based program designed UAV applications, that can work even in rough scenarios, like in almost no light conditions and in very low textured environments. An optimization-based framework, VINS-Fusion, was also considered for its good results in VIO-SLAM mode and its VIO-only module was tested. The last framework considered was OKVIS, also optimization-based, but older then VINS. Its main quality is the use of BRISK to extract robust features and improve the localization accuracy.

1.1 Objectives

The main objectives consist in:

- Analyze the camera factory calibration and re-calibrate it if needed,
- Define a test methodology and a metrics to compare the algorithms,
- Determine the best camera settings and parameters settings for each framework, and confront their accuracy in an indoor environment.

Chapter 2

Stereo camera

A stereo camera is a combination of two cameras placed at known distance, with a separate image sensor for each lens. They are used because of the possibility to reconstruct a 3D scene via stereo triangulation.

2.1 MYNTEYE S1030-IR-120/Mono

The MYNTEYE S1030-IR-120/Mono is a monochromatic visible and infrared (IR) stereo camera, with an integrated 6-axis Inertial Measurement Unit (IMU) and a high visual angle [1]. It has a global shutter and the cameras are hardware synchronized. The manufacturer does not specify if the cameras measurements are synchronized with the IMU measurement with and hardware trigger, so test were performed. The produce provides a software development kit (SDK) that was used on Ubuntu 16.04 operative system to interface with the camera and record data using the given ROS packages. The Robot Operating System (ROS) is "an open-source, meta-operating system" for robots. It provides the services that an operative system usually does and a set of tools to run code in multiple machines [2]. To test the camera-IMU synchronization it was used both a program present in the camera software development kit (SDK) and a self-made script to analyze data recorded with ROS. The results show no hardware triggering there is instead a big fluctuation in the closest time difference between camera e IMU measurements. The oscillation is centered in zero and with an amplitude (2 times the maximum time difference) that depends on the IMU frequency settings, while its frequency depends on the camera fps settings, being about half of it. The behavior is shown in table 2.1

2.2 Camera calibration

Geometric camera calibration is a composition of the camera modeling, that deals with a mathematical approximation of the optical physics using a set of parameters, and the calibration, that deals with the use of direct or iterative methods to estimate the values of these parameters [3].

IMU frequency [Hz]	Maximum time difference [ms]
100	± 5.0
200	± 2.5
250	± 2.0
333	± 1.5
500	± 1.0

Table 2.1: MYNTEYE S camera-IMU asynchrony.

Model	S1030-IR-120/Mono
Size	PCB dimension 149mm × 24 mm total dimension 165×31.5×29.6 mm
Frame Rate	10/15/20/25/30/35/40/45/50/55/60 FPS
Resolving Power	752x480
Depth Resolving Power	Base on CPU/GPU Up to 752×480@60FPS
IR detectable range	Up to 3m
Visual Angle	D:140°H:120°V:75°
Color Pattern	Monochrome
Pixel Size	6.0 × 6.0 μm
Shutter Speed	17 milliseconds
Baseline	120.0 mm
Camera Lens	M6 Camera Attachment
Focal Length	2.1mm
Motion Perception	6 Axis IMU
Control Function	Exposure/Shutter/Brightness/IR
IMU Frequency	100/200/250/333/500 Hz
Cameras Synchronization Precision	<1ms (up to 0.05ms)
Working Distance	0.8-5m+
Scanning Mode	Global shutter
Power	3.5W @ 5V DC from USB
Output data format	Raw data
Data transfer Interface	USB 3.0
Weight	196g
UVC MODE	YES

Table 2.2: MYNTEYE S1030-IR datasheet [1].

FIGURE 2.1: MYNTEYE S camera [1].

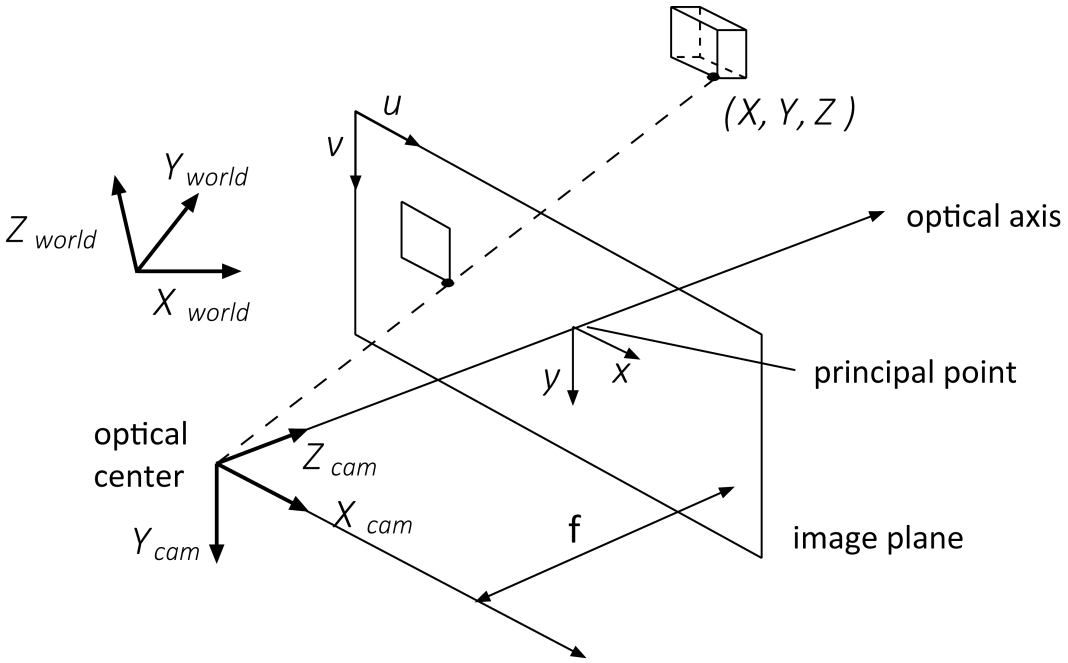


FIGURE 2.2: Camera projection model .Image adapeted from [3].

2.2.1 Extrinsic and intrinsic camera model

The extrinsic camera model provides a transformation of the camera position and orientation with respect to a world coordinate system. This is performed by a homogeneous transformation matrix that represents a roto-translation ([4], [5]):

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}_{cam} = \left[\begin{array}{ccc|c} & \mathbf{R}_w^c & & \mathbf{t}_w^c \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}_{world}, \tag{2.1}$$

where $\mathbf{t}_w^c$ is a vector that represents the origin of the world reference frame with respect to the camera reference frame, and $\mathbf{R}_w^c$ is a rotation matrix that represents the axes of the world reference frame in the camera one.
The intrinsic camera model describes the transformation from a 3D point in the camera

frame to a 2D point in the image frame and therefore models the internal geometry and optical characteristics of the image sensor [3]. It is usually composed by a projection model and a lens distortion model.

2.2.2 Pinhole projection model

As noted in [5] and [6], the pinhole model assumes that all the light rays pass through a single point, the optical center or camera center, and that there is a linear correlation between the image point position and the ray direction. This transforms a 3D point in the camera frame in a 2D point in the image frame, using prospective projection. The intersection between the camera optical axis and the image plane is called principal point. In the model, the image plane is moved in front of the principal plane or lens plane, to avoid the image inversion (and becomes a virtual image plane), as seen in fig.2.3. The geometric mapping from 3D to 2D can be expressed, using homogeneous

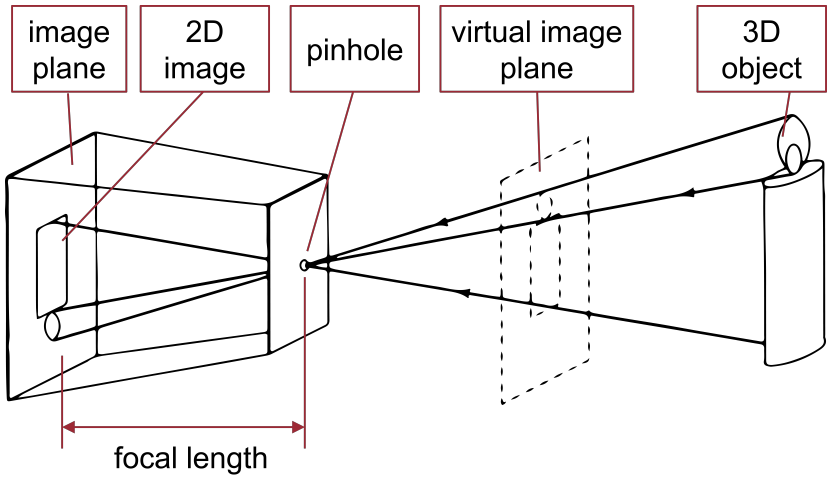


FIGURE 2.3: Pinhole projection model [6].

coordinates, as

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \\ 1 \end{bmatrix}, \tag{2.2}$$

where λ is the homogeneous scaling factor, f is the focal length (in pixels) and the 3×4 matrix is called the projection matrix ($\mathbf{P}$). Most of the current cameras define the pixel coordinate system origin at the top left corner of the image. Therefore the coordinates of the principal point are $[u_0\ v_0]^T$. This changes the prospective projection equation as

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & u_0 & 0 \\ 0 & f & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \\ 1 \end{bmatrix} \tag{2.3}$$

The real sensors have more imperfections that must be taken in account. In fact, the pixels might not be squared, leading to two different focal lengths in each direction, and can potentially be skewed, leading to a skew factor s. The pixel can appear skewed in case that the image is acquired by a frame grabber, due to an inaccurate synchronization of the pixel-sampling process. The more general definition of the projection mapping is then:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} & & \\ & \mathbf{K} & \\ & & \end{bmatrix} \begin{array}{|c|} \mathbf{0} \end{array} \begin{bmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \\ 1 \end{bmatrix} \tag{2.4}$$

The matrix K is called intrinsic camera matrix and

$$\mathbf{K} = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

In practice, using recent digital camera, we can assume that $f_x = f_y$ and $s = 0$.

2.2.3 Lens distortion models

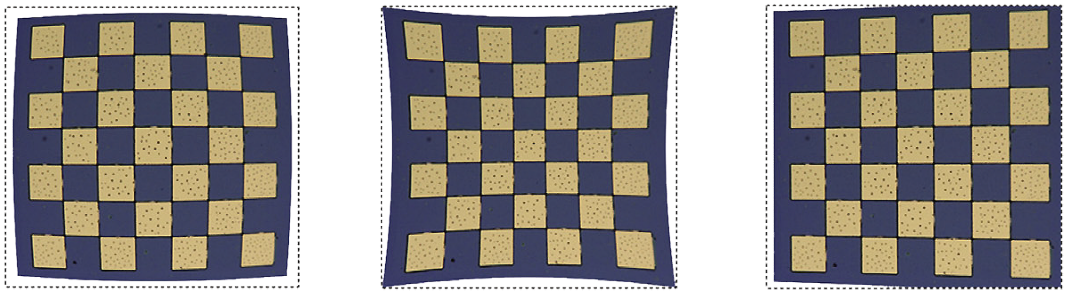
In real cameras there is a non-linear distortion produced by the lenses, that we have to take in account [7]. Consider the pinhole projection at unitary focus $x = \frac{X_{cam}}{Z_{cam}}$, $y = \frac{Y_{cam}}{Z_{cam}}$ and $r = \sqrt{x^2 + y^2}$ Of course then, considering the distorted pixel coordinates, $[x_{dist}, y_{dist}]^T$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} & & \\ & \mathbf{K} & \\ & & \end{bmatrix} \begin{bmatrix} x_{dist} \\ y_{dist} \\ 1 \end{bmatrix} \tag{2.6}$$

Radial-tangential distortion model

For the cameras with low field of view, a good model is the one considered in OpenCV, [8], the radial-tangential distortion model. OpenCV is a open-source vision library wildly used for real time image processing. The aforementioned model needs 5 parameters, $(k_1, k_2, p_1, p_2, k_3)$. As stated in [10], "radial distortion is mostly cause by a flawed radial curvature curve of the lens elements". It depends on the radial distance from the principal point of the image plane, so it is most noticeable at the image periphery. A barrel distortion consists in a negative radial displacement of the image points, and it causes outer points to get increasingly closer to each other and the scale to decrease (fig .2.4). A pincushion distortion is instead a positive radial displacement that causes outer points to spread and the scale to increase.
The function used to model this OpenCV is

$$\begin{aligned} x_{dist}(x) &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_{dist}(y) &= y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{aligned} \tag{2.7}$$



(A) Barrel distortion (B) Pincushion distortion (C) Tangential distortion

FIGURE 2.4: Radial-tangential distortions [9].

The tangential distortion depends mostly on the non-parallelism between the image plane and the lens plane. It is an asymmetrical effect and is much smaller compared with radial distortions for modern cameras.

$$\begin{aligned} x_{dist}(x,y) &= x + 2p_1xy + p_2(r^2 + 2x^2) \\ y_{dist}(x,y) &= y + p_1(r^2 + yx^2) + 2p_2xy \end{aligned} \tag{2.8}$$

Of course the total distortion is the sum of the two.

Kannala-Bradt distortion model

Both OpenCV, Matlab and Kalibr have the same camera model, called fisheye in the first two and equidistant in the third. This is a combination of a pinhole camera model with a Kannala-Bradt distortion model. Even if this is not one of the full camera models for fisheye lenses as in [11], it is a good enough simplification. The equidistant name, comes from the equidistant camera model it is based of. The equidistant camera model has a projection model parameterized as $r = f\theta$, where θ is the angle between the principal axis and the incoming ray, r is the distance between the image point and the principal point and f is the focal length (fig. 2.5) [11]. Most of the fisheye lenses are made to follow this projection model, this is way it is the most used in calibration toolboxes. Using the projection model as a distortion model, we consider the distortion given by a sphere projecting on a plane. Considering $\theta = \arctan(r/Z_{cam})$, the Kannala-Bradt distortion model is defined as [12]:

$$\theta_{dist} = \theta + k_1\theta^3 + k_2\theta^5 + k_3\theta^7 + k_4\theta^9, \tag{2.9}$$

and the distortion pixel coordinate become

$$\begin{aligned} x_{dist}(x,y,Z_{cam}) &= \theta_{dist}\frac{x}{r} \\ y_{dist}(x,y,Z_{cam}) &= \theta_{dist}\frac{y}{r}. \end{aligned} \tag{2.10}$$

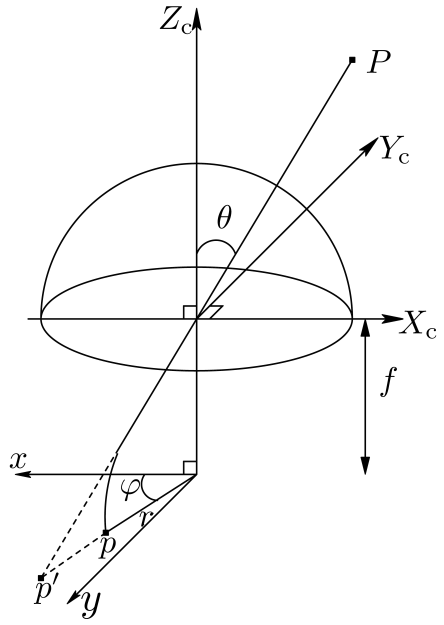


FIGURE 2.5: Fisheye camera model. The image of the point P is p, whereas it would be $p^{'}$ using a pinhole camera model [11].

2.2.4 Best camera calibration practices

As suggested in [13] and [14] the best calibration practices are the following.

1. Camera focus: set the camera focus to manual and at the working distance of the final application, at which also the calibration is going to be performed,

2. Vibrations and blur: mechanically lock the camera to keep the vibrations as low as possible. This reduces the blur, making the feature detection easy and accurate. In case of large targets, consider mounting them vertically or laying flat on a rigid support and move the camera instead,

3. Pattern:

   - Pattern size: it should preferably cover half of the total field of view when seen parallel to the camera sensor,
   - Pattern borders: leave a white margin as big as the distance between two inner corners, to allow the feature detection algorithm to work well and faster.
   - Pattern feature density: it should have high feature count, but at some point, the detection robustness suffers. The recommendation from [13] is to use fine pattern counts for cameras above 3MPx and if the lighting is controlled and good,
   - Pattern manufacturing quality: the calibration is only accurate as the calibration target used, so use laser printed patterns only to validate and test. Print it on a white opaque, planar and rigid material, making sure it is as

flat as possible. Check the laser printed ones for scaling during the printing process,

- Pattern type: targets with unique coding as CharuCo boards or Aprigrids (fig 2.6a) are very good because they allow to record valid data even at edge of the camera field of view and with partial views. Moreover, their rotation is detectable.
  Circular patterns can be potentially more accurate than checkerboards, but only if there is a proper sub-pixel determination of the center of the circles. This is sensitive to a proper choice of circle sizes and the math regarding corrections is more complex [15],

4. Lightning: diffuse lightning is better, preferably a controlled photographic one. Strong point sources create uneven illumination that can result in detection fail,

5. Image collection: the target should be moved to cover all the camera field of view with and even distribution. It is also important to tilt the target in both horizontal and vertical direction up to $\pm$ 45 degrees, because the lens distortion can be recovered just with images parallel to the view, but the focal length estimation needs different depths. Usually 10 observations should be a good number,

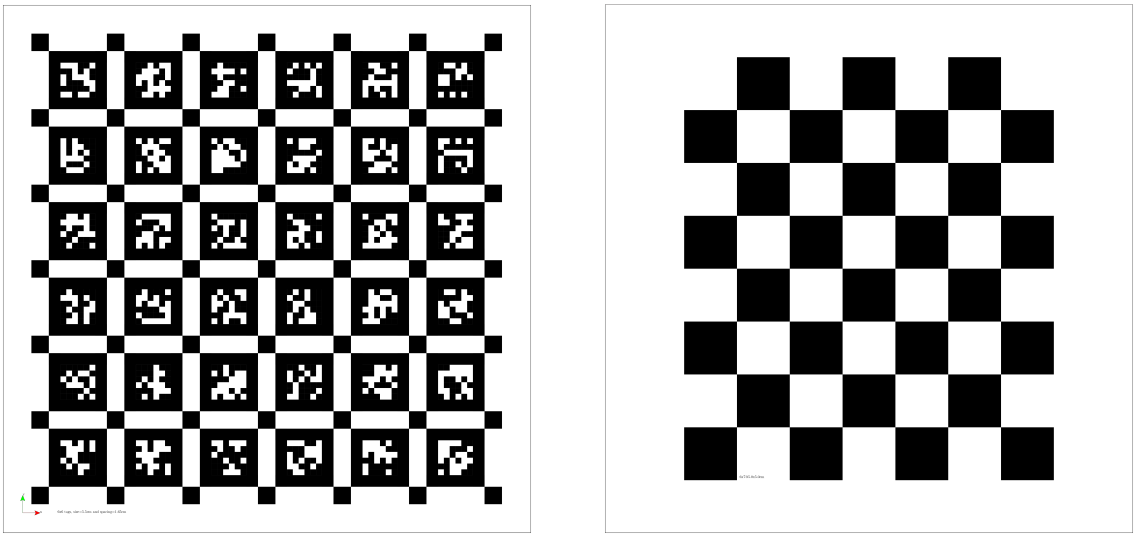
6. Post-processing: carefully inspect reprojection errors, both per-view and per-feature. If any of these appear as outliers, exclude them and re-calibrate,

7. Result analysis: the calibration algorithm tries to minimize the reprojection error. A good value is less than 0.5 pixel. However, a low mean squared value only means that the algorithm was able to minimize the object function on average. A good approach is to manually measure, for every inner feature, the reprojection error and plot the results in a histogram, that should be centered at zero.

2.3 Camera calibration with Kalibr

Kalibr is a open-source toolbox that performs multiple camera extrinsic and intrinsic calibration, and camera-IMU spatial and temporal calibration [16]. Its used was proposed in the MYNTEYE SDK guide [17] and it was chosen due to several features [16], [18]:

- ROS integration: it uses ROS bag files as input data and it has tools that work only in this environment,
- Automatic: it does not require user intervention during the procedure,
- Flexibility: supports 4 camera projection models, 4 distortion models and 3 calibration targets,
- Robust feature extraction and optimization algorithms,



(A) 6x6 Aprilgrid (B) 7x6 Checkerboard

FIGURE 2.6: Calibration target. [19]

- Performs camera-IMU calibration (which most calibration toolboxes do not),
- Provides error metrics and respective data visualization,

2.3.1 Camera calibration procedure used

Since a pattern half of the camera field of view would have been huge and there are not many indications about its dimensions in the literature, the first try was with an A4 sheet. It was printed on it a scaled 6x6 Aprilgrid of 0.5x0.5 m on an A0 (fig. 2.6a). The use of this target lead to unacceptable reprojection errors. Therefore, the second try was with one of the standard calibration targets found on the Kalibr wiki [19], composed by a 6x6 Aprilgrid and a 7x6 Checkerboard both 0.5x0.5 m big, on an A0 sheet (fig. 2.6). The sheet was cut in half, each target was taped on one side of a wooden board, by making sure they were as flat as possible. The tape was used, since it was less of a permanent solution the glue, thus allowing to easily change the target type in the future.
From many trials it was discovered that the checkerboard gives lower reprojection errors, so it was used for the stereo camera calibration. On the contrary, the Aprilgrid gives higher identification rates with fast moving camera, so it was used for the camera-IMU calibration. In order to limit strong point source illumination, it was used a combination of natural and artificial lightning, and the target was oriented correctly with respect to the light sources .

- Camera calibration: the camera was fixed on a tripod and the pattern was kept further then the guidelines, as it covered 1/8th of the field of view. The target was moved in a manner that displayed always about the same dimension in the undistorted image (i.e. in a arc at about 1.5 m from the camera). The pattern

was tilted following the guidelines, but sopping at a lower angle of about 30 degrees, since Kalibr could not detect any corners for higher inclinations. The camera field of view was covered evenly moving the target slowly (on average it took about 1 minute and 45 second).

- Camera-imu calibration: the calibration target was fixed laying it against the wall and, staying closer than the previous case, the camera was moved as shown in the video on the Kabir Wiki [19] (translation and rotation along each axis separately, followed by a random movement). In this phase Kalibr can provide an online estimation of the camera-IMU measurement delay. This function was turn of because it gave wrong estimations, (e.g 0.0067 while it should be closer to 0 s 2.1).

2.3.2 Kalibr setting and result analysis

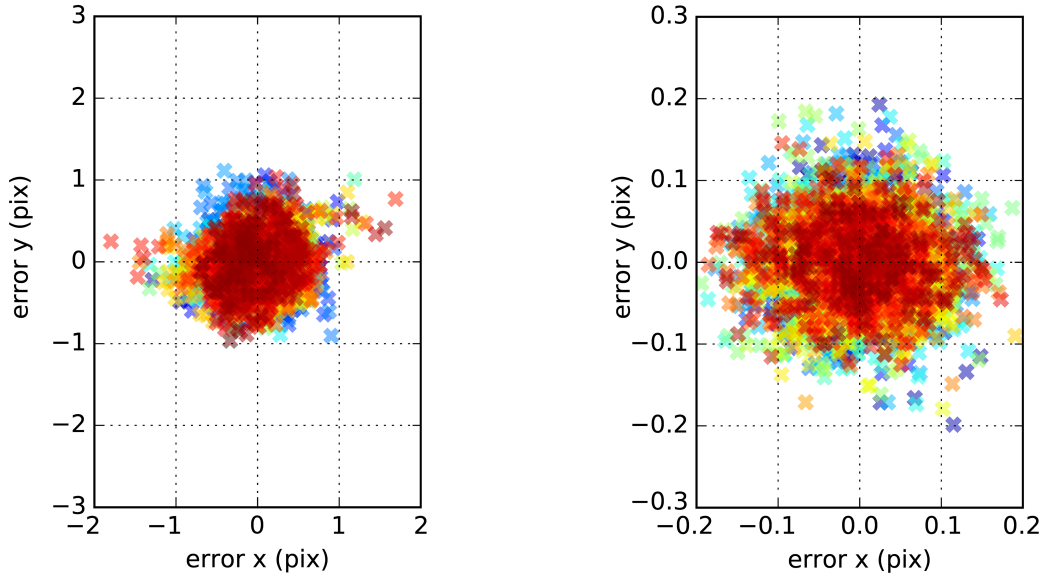
Since the camera came pre-calibrated with a pinhole projection model and a radial-tangential distortion model, the first calibration trials were preformed using the same models. It was not possible to get reprojection errors lower than 0.5 pixels (fig. 2.7a), and after checking with the factory calibration, it was discovered that it had the same issue. The camera field of view (tab. 2.2) leads the image to be too distorted for the radial-tangential model. It was therefore tried the Kannala-Bradt distortion model, that lead to far lower reprojection errors, as shown in fig. 2.7b. The best calibration results were chosen looking at the field of view coverage, at the reprojection error and the accuracy of the estimation of the known parameters (the camera baseline). For the camera-imu calibration, since there are not known parameters currently declared by the manufacturer, it was used the data taken from a old version of the SDK guide (more then one year old) found at [52] and checked if the results were similar.

2.4 Stereo triangulation

A stereo camera it is used because, as for human's eyes, once a point position is know in two views, it is possible to reconstruct its 3D coordinates. This operation is called triangulation and in ordered to easily find the pair of 2D points describing a 3D point, it is important to have a calibrated stereo camera and transform the images. This requires applications of epipolar geometry.

2.4.1 Epipolar geometry

As noted in [20], epipolar geometry relates a 3D point and the corresponding observations in a stereo camera setting. Consider a pair of cameras with camera centers are O_l and O_r. The line between them is called baseline (usually the norm of the baseline is referred at with the same name) and the plane defined by the camera centers and the point P is called epipolar plane. The points at which the baseline intersect the camera planes are called epipols and the line defined by the intersection of the epipolar plane



(A) Radial-tangential distortion model

(B) Kannala-Bradt distortion model

FIGURE 2.7: Confront between reprojection errors. The results come from the same dataset and the errors are in pixels. The different colors indicate different images.

and the two image planes are known as the epipolar lines. The epipolar lines intersect the baseline at the respective epipoles in the image plane. If the projection of the point P on the right image plane is known, since the epipolar plane is defined, the projection of the same point on the left image plane must lie on the analogous epipolar line. This property, called epipolar constraint, for a calibrated camera is written as $\mathbf{p}_l^T \mathbf{E} \mathbf{p}_r = 0$, where $\mathbf{p}$ are 2D homogeneous vectors and $\mathbf{E} = \mathbf{t}^{\times}\mathbf{R}$, with $\mathbf{t} = \mathbf{t}_{\mathcal{C}_r}^{\mathcal{C}_l}$ the translation vector between left and right camera frame and $\mathbf{R} = \mathbf{R}_{\mathcal{C}_r}^{\mathcal{C}_l}$ the analogous rotation matrix (fig .2.8).

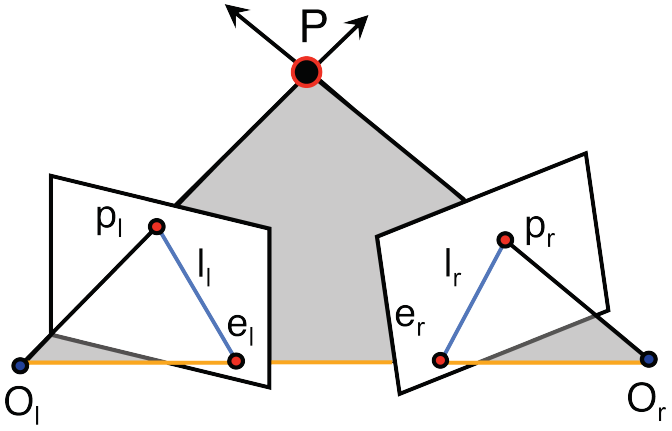


FIGURE 2.8: Epipolar geometry model in which are shown the camera centers O, the epipoles e, the point P projections on the image frames and the epoipolar lines l[20].

2.4.2 Stereo triangulation

As suggested in [21], using a calibrated camera it is possible to rectify the image pair, making the epipolar lines horizontal and the corresponding image ones in both images collinear. This requires undistorting the images and using the essential matrix to make the image plane complanar. This limits the search of a point correspondence, known the position on the other camera image plane, from 2D to 1D, only concerning the line with $v = v_p$. Different algorithms can be used to find the match, but one it is found the point disparity can be computed $d = u_r - u_l$. From the disparity the point depth can be found. In fact, considering the triangles similarity, it is easy to prove that $\frac{b}{Z_p} = \frac{(b + u_r) - u_l}{Z_p - f_x} \rightarrow Z_p = \frac{f_x b}{d}$ (fig . 2.9).

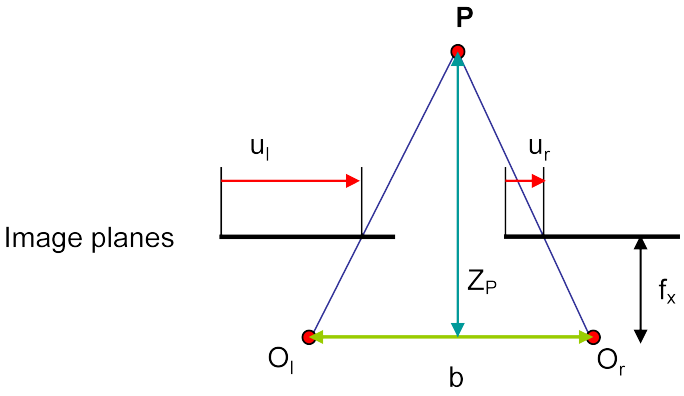


FIGURE 2.9: Stereo triangulation [21].

2.4.3 Disparity space

As explained in [22], the disparity space is used to express stereo-triangulated points using their u, v coordinates on the image plane and disparity d. Considering a pair of matching points $[u_r \ v_r]^T, [u_l \ v_l]^T$, the transformation to the disparity space is defined as:

$$\mathbf{y} = \begin{bmatrix} u \\ v \\ d \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 1 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} u_l \\ v_l \\ u_r \\ v_r \end{bmatrix}. \tag{2.11}$$

Considering a calibrated stereo camera, the following map from the homogeneous coordinate space to the disparity space is defined:

$$\mathbf{y} = \mathbf{g}(\mathbf{p}) \rightarrow \begin{bmatrix} u \\ v \\ d \end{bmatrix} = \frac{1}{Z_p} \begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 0 & bf_x \end{bmatrix} \begin{bmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{bmatrix} \tag{2.12}$$

where b is the baseline norm and $\mathbf{g}(\cdot)$ is called the stereo projection model.

Chapter 3

Elements of visual odometry

As shown in greater details in [23] and [24], visual odometry (VO) is defined as the set of algorithms that provide and incremental online estimation of an agent's (e.g. vehicle, robot) position by analyzing the image sequences captured by a camera. The idea of estimating a vehicle's pose from visual input alone starts in 1960's, where Stanford University built a lunar rover controlled from the Earth. This design was further investigated by Moravec to demonstrate correspondence based stereo navigation in 1980s. NASA monopolized VO research From 1980 to 2000, when the technologies for the 2004 Mars Mission were being developed. The term "visual odometry" was invented because of the similarities between vision-based localization and wheel odometry. In fact, as the latter integrates incrementally the number of a vehicle wheels turns over time to estimate its movement, VO integrates pixel displacements between image frames over time.

3.1 VO approaches

The use of vision-based odometry to estimate the position of a mobile robot can be categorized in many different ways as detailed in [23] and [24]. Focusing on the model used to compute the movement, it can be divided in geometric approaches, that apply concepts from projective geometry , and non-geometric approaches that use learning algorithms for the estimation. The geometric approaches can be subdivided considering the type of exploited visual information in: feature based, appearance based and hybrid approaches.

3.1.1 Feature based approach

This strategy involves the extraction of features (usually corners, sometimes lines) in a sequence of images, and matching or tracking a subset of them in all the images, to estimate the camera motion. Traditional edge and corner feature detectors, as Moravec and Harris corner detectors, provide a really fast image correspondence, but suffer of low robustness. Nowadays, scale and transformation invariant feature extraction

algorithms as SIFT, SURF, ORB, BRISK are widely used. They work by assigning to each extracted corner a descriptor, that increases the matching speed and robustness. The selection of a good subset of features is really important since it can lower the computational burden and increase the estimation accuracy. One simple method is to use a bucketing technique to evenly distribute features in the image. More complex ones involve the recognition and use only of static features (e.g the features belonging to a door frame). It is also paramount to implement some outlier removal technique as RANSAC or based on a model of the motion. If stereo VO is implemented, the extracted features from the first frame are matched with the corresponding points in the second frame, thus providing the 3D position of the points in space.

3.1.2 Appearance based approach

The image information is used directly, by the optimization of the photometric error for motion estimation. This error takes in account the change in appearance of acquired images and the intensity of pixel information. The feature-based techniques are usually noisy and the features can not be extracted in smoothly varying landscapes (e.g foggy environment). They also suffer from association mismatch with similar looking places. Using whole images reduces these issues, but makes the system less robust to noise. This techniques can be divided in: region-based matching and optical flow based.

- *Region-based matching*: The region-based matching either can be achieved through correlation (template matching) or with the use of global appearance based descriptors and image alignment approaches. They relay on the definition of a specific area of interest and require a minimization of an objective function with an optimization method. A incorrect choice of this function can lead to local minima a divergence issues. In more details the techniques consist in:
  - Template matching: is the most commonly used method and consist in selecting a patch or a template from the current image frame and attempting to match this patch in the next frame. In this approach the template is shifted around a new acquired image frame and a similarity degree is computed by different measures. The area where this value is higher is most likely the new position of the template. Two similarity scores are most commonly used: sum of squared differences (SSD) and normalized cross correlation (NCC). NCC is more accurate, but also slower in execution.
  - Global appearance descriptors: Fourier-Mellin transformations, Fourier image signatures (zero phase representation included) and image histograms are the most used. Each on has its pros and cons, and overall the choice depends on the environment in which the images are taken and the computational power of the system. It is very important to optimized their parameters because the computation burden can become unbearable very easily.

- *Optical flow matching*: it does not depend on a definition of a objective function and can better tolerate moving objects, then the region based techniques. Its genesis comes from the observation of the insects' biology. The algorithm assumes that the pixels intensity only shifts in a limited time frame without changing value. If all image pixels are used to estimate the optical flow the algorithms are called dense optical flow estimators. When only a selected number of pixels are used the technique is named sparse optical flow.

- *Hybrid of feature and appearance based approaches*: feature-based approach better preforms in textured environments, while appearance-based approach is more robust in low-textured scenarios. The combination of the two is the best solution in some scenarios.

3.1.3 Image pyramids

As noted in [25] and [26], it is a data structure formed by a set of images at multiples scales. Its use in computer vision is motivated mainly by the reduction of computation cost given by their used in feature detection algorithms. Their use also extends to image enhancement and analysis algorithms.
The pyramid zero level consists of the original image. Then, each new layer is obtained iteratively from the previous one, applying a smoothing filter and reducing the image size by a downsampling. A scale factor is imposed which regulates the filter intensity and the scale factor.
Different kernels (e.g. Gaussian, Laplacian) can be convoluted with the image based on the intended application. A visual representation of this data structure is depicted in figure 3.1.

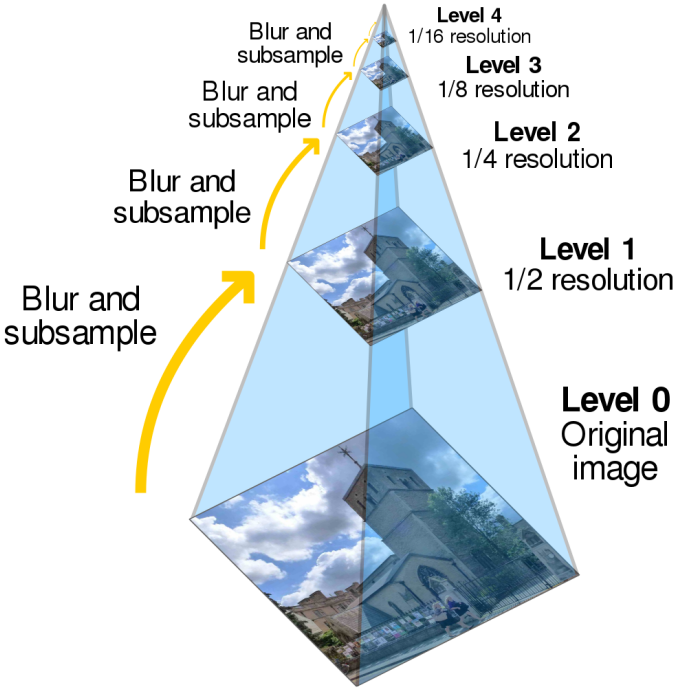


FIGURE 3.1: Representation of an image pyramid with 5 levels and 2 scale [27].

3.2 FAST: Features from Accelerated Segment Test

As explained in [28], the FAST detector returns an high number of corner features in a short time. It works considering a circle of sixteen pixels around the corner candidate p with pixel intensity I_p (fig. 3.2). The point p is classified as corner if one of this conditions is respected:

- There exists a set of n contiguous pixels S in the circle which, $\forall x \in S$, the intensity of x, $I_x > I_p + t$,
- There exists a set of n contiguous pixels S in the circle which, $\forall x \in S$, $I_x < Ip + t$

where t is a threshold, usually set at 12, since it admits a high-speed test which can be used to exclude a very large number of non-corners. This consists in examining only the pixels at the four cardinal directions (1, 5, 9 and 13). If p is a corner, then at least three of these pixels must be brighter than $Ip + t$ or darker than $Ip + t$. If p does not belong to any of these occurrences, it cannot be a corner. With this method the full test is applied only on the corner candidates, examining all pixels in the circle. This detector exhibits high performance, but there are several weaknesses:

- The high-speed test does not generalize well for n < 12,
- The choice of 4 pixels in a specific order represent an assumption about the appearance of a feature,

- The information about relative pixel brightness from the first 4 tests is not used in the full test ,
- Features are not well spaced in the image frame.

To address the first three problems it is used machine learning, while for the last it is applied non-maximal suppression.

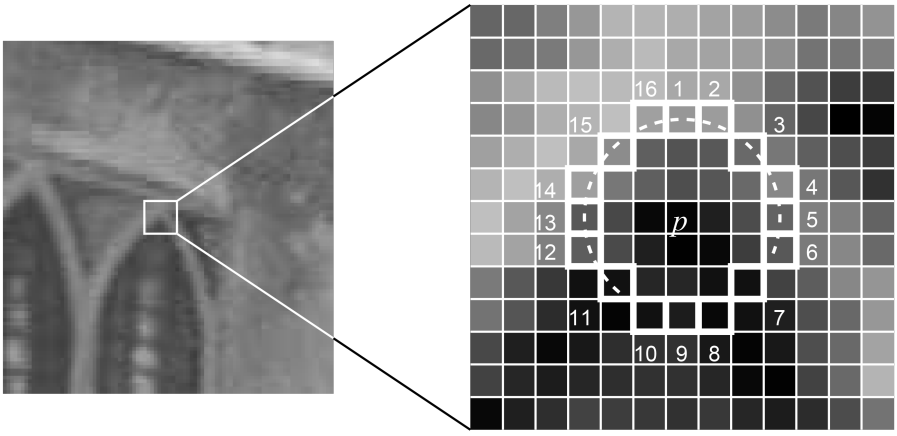


FIGURE 3.2: FAST algorithm implementation. The pixel p is the center of a candidate corner. The arc is indicated by the dashed line and passes through 12 contiguous pixels which are brighter than p by more than the threshold. The highlighted squares are the pixels used in the corner detection [28].

3.3 Harris corner detector

As argued in [29] and [30], it is a standard technique to locate interest points on a image. The method is based on illumination changes therefore it is applied on grayscale images. The basic idea is to detect points based on the intensity variation in a local neighborhood of a feature. So, the pixel intensity should vary more in a small region around the feature than in any windows shifted in any direction. The image is defined as a function $I : \Omega \in \mathbb{R}^2 \to \mathbb{R}$ and h is a small increment from any point belonging to the following domain: $\mathbf{x} = [x\ y]^T \in \Omega$. Corners are defined as the points x that maximize the following functional:

$$E(\mathbf{h}) = \sum_{\mathbf{x}} w(\mathbf{x})(I(\mathbf{x}+\mathbf{h}) - I(\mathbf{x}))^2, \tag{3.1}$$

so, that give the maximum intensity variation in any direction. The function $w(\mathbf{x})$ defines a support region to minimize the aliasing effects. It is usually a rectangular or Gaussian function. In order to speed up the computation a Taylor expansion is used:

$I(\mathbf{x}+\mathbf{h}) \approx I(\mathbf{x}) + \nabla I(\mathbf{x})^T\mathbf{h}$ resulting in:

$$E(\mathbf{h}) \approx \sum_{\mathbf{x}} w(\mathbf{x})(\nabla I(\mathbf{x})^T\mathbf{h})^2 = \sum_{\mathbf{x}} w(\mathbf{x})(\mathbf{h}^T\nabla I(\mathbf{x})\nabla I(\mathbf{x})^T\mathbf{h}) = \mathbf{h}^T\mathbf{M}\mathbf{h}. \tag{3.2}$$

$\mathbf{M}$ is the autocorreletion matrix, defined as:

$$\mathbf{M} = \begin{bmatrix} \sum_{\mathbf{x}} w(\mathbf{x})I_x^2 & \sum_{\mathbf{x}} w(\mathbf{x})I_xI_y \\ \sum_{\mathbf{x}} w(\mathbf{x})I_xI_y & \sum_{\mathbf{x}} w(\mathbf{x})I_y^2 \end{bmatrix}, \tag{3.3}$$

with $I_* = \dfrac{\partial I(\mathbf{x})}{\partial *}$. The maxima of $E(\mathbf{h})$ are found analyzing matrix $\mathbf{M}$. Its largest eigenvalue correspond to the direction of largest intensity variation, while the second one corresponds to the intensity variation in its orthogonal direction. Three cases may present:

- Both eigenvalues are small, $\lambda_1 \approx \lambda_2 \approx 0$, the area is most likely to lack of features,
- One of the eigenvalues is much larger than the other one, $\lambda_1 \gg \lambda_2 \approx 0$, then the region is likely to belong to an edge,
- Both eigenvalues are large, $\lambda_1 > \lambda_2 \gg 0$, so the area contains variations in both orthogonal direction, characterizing a corner.

In the actual implementation, autocorrelation matrix is symmetric and positive semi-definite, so it can be defined a corner response function that is invariant to in-plane rotations.

3.3.1 Harris measure

The measure proposed by Harris and Stephens is:

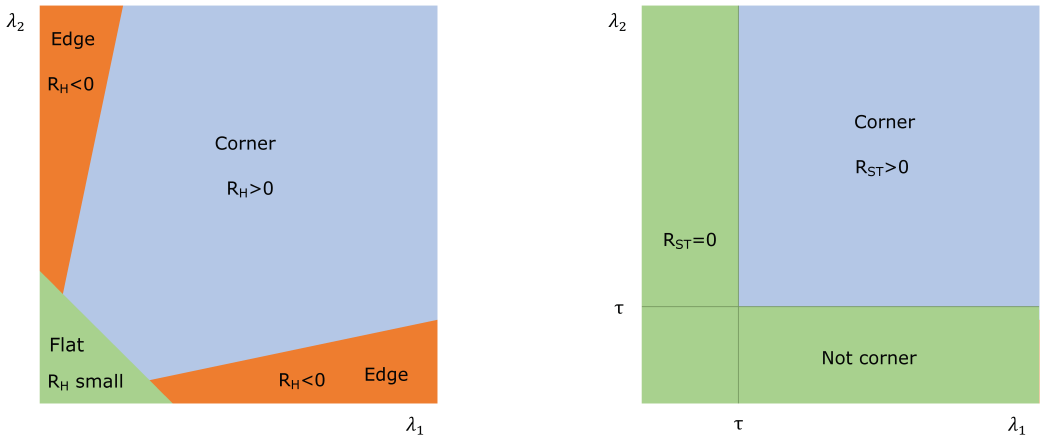
$$R_H = \lambda_1\lambda_2 - k(\lambda_1+\lambda_2)^2 = \det(\mathbf{M}) - k\ \mathrm{trace}(\mathbf{M})^2, \tag{3.4}$$

where usually $k = 0.04 : 0.06$. This function allows to detect corners and edges. As shown in the figure 3.3a, when R is negative, one of the eigenvalues is much larger than the other, thus the pixel is likely to belong to an edge. When R is positive and large, both eigenvalues are large, then it is likely to be a corner. If both eigenvalues are small, R is small and it is part of a flat region.

3.3.2 Shi-Tomasi measure

The corner response function is

$$R_{ST} = \begin{cases} \lambda_{min} & \text{if } \lambda_{min} > \tau \\ 0 & \text{otherwise} \end{cases}, \tag{3.5}$$



(A) Harris measure [29]

(B) Shi-Tomasi measure. Image adapted from [31]

FIGURE 3.3: Harris and Shi-Tomasi measures. In the figures are depicted areas with different values of the corner response function R

where $\lambda_{min} = \min(\lambda_1, \lambda_2)$. In this case if the minimum eigenvalue it is greater than the threshold, the point is considered a corner (fig. 3.3b).

3.4 KLT feature tracker

As explicate in [32], it consists in a sparse optical flow based visual odometry technique. Defined $I(x, y)$ as the intensity function of a image frame and $J(x, y)$ of another image frame (as in 3.3). If $\mathbf{u} = [u_x \; u_y]^T$ the point to be tracked on the first image, the goal is to find $\mathbf{v}$ on J where $I(\mathbf{u})$ and $J(\mathbf{v})$ are similar. $\mathbf{v}$ is defined as $\mathbf{v} = \mathbf{u} + \mathbf{d}$ where $\mathbf{d}$ is the image velocity at $\mathbf{u}$, also called optical flow. This is equivalent to minimize the residual function $\varepsilon(\mathbf{d})$, defined as:

$$\varepsilon(\mathbf{d}) = \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} \left(I(x, y) - J(x + d_x, y + d_y)\right)^2, \tag{3.6}$$

where it is considered a rectangular window of size $(2\omega_x + 1) \times (2\omega_y + 1)$. Since the standard implementation of the KLT algorithm can only deal with small pixel displacement, it is usually realized with a pyramidal representation (sec. 3.1.3). Starting from the layer 0 each layer is downscaled by a factor 2. So if $\mathbf{u} := \mathbf{u}^0$, $\mathbf{u}^L = \frac{\mathbf{u}}{2^L}$. The algorithm works computing d^L at the L pyramid level, that is scaled and used as first guess to the lower layer until the layer 0 is reached. This is done by minimizing the error function ε^L defined as:

$$\varepsilon^L(\mathbf{d}^L) = \sum_{x=u_x^L-\omega_x}^{u_x^L+\omega_x} \sum_{y=u_y^L-\omega_y}^{u_y^L+\omega_y} \left(I^L(x, y) - J^L(x + g_x^L + d_x^L, y + g_y^L + d_y^L)\right)^2, \tag{3.7}$$

in which the second image is pre-translated by $\mathbf{g}^L = 2(\mathbf{g}^{L+1} + \mathbf{d}^{L+1})$ that is the lower level optical flow. Of course $\mathbf{g}^{L_m} = [0\ 0]^T$. This implementation keeps $\mathbf{d}^L$ small and easy to compute. The final optical flow is:

$$\mathbf{d} = \sum_{L=0}^{L_m} 2^L \mathbf{d}^L. \tag{3.8}$$

3.4.1 Standard implementation

The goal is to find $\mathbf{d}_L$ that minimizes the function ε^L, for all levels. To generalize the notation the point vector is $\mathbf{p} = \mathbf{u}^L$, the displacement vector is $\hat{\boldsymbol{\nu}} = \mathbf{d}^L$ and $A(x,y) := I^L(x,y)$ and $B := I^L(x + g_x^L, y + g_y^L)$. This transforms the goal in finding the $\hat{\boldsymbol{\nu}}$ that minimizes:

$$\varepsilon(\hat{\boldsymbol{\nu}}) = \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \left(A(x,y) - B(x+\nu_x, y+\nu_y)\right)^2, \tag{3.9}$$

In order to find the optimum $\left.\dfrac{\partial \varepsilon(\hat{\boldsymbol{\nu}})}{\partial \hat{\boldsymbol{\nu}}}\right|_{\hat{\boldsymbol{\nu}}=\hat{\boldsymbol{\nu}}_{opt}} = [0\ 0]$. And

$$\frac{\partial \varepsilon(\hat{\boldsymbol{\nu}})}{\partial \hat{\boldsymbol{\nu}}} = -2 \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \left(\left(A(x,y) - B(x+\nu_x, y+\nu_y)\right) \begin{bmatrix} \dfrac{\partial B}{\partial x} & \dfrac{\partial B}{\partial y} \end{bmatrix} \right). \tag{3.10}$$

Substituting $B(x+\nu_x, y+\nu_y)$ with its 1st order Taylor expansion about the point $\hat{\boldsymbol{\nu}} = [0\ 0]^T$,

$$\frac{\partial \varepsilon(\hat{\boldsymbol{\nu}})}{\partial \hat{\boldsymbol{\nu}}} \approx -2 \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \left(\left(\delta I - \nabla J^T \hat{\boldsymbol{\nu}}\right) \nabla J^T \right). \tag{3.11}$$

where the frame difference is $\delta I := A - B$ and the second image gradient is $\nabla J = \begin{bmatrix} J_x \\ J_y \end{bmatrix} := \begin{bmatrix} \dfrac{\partial B}{\partial x} & \dfrac{\partial B}{\partial y} \end{bmatrix}^T$. This means that

$$\frac{1}{2}\left[\frac{\partial \varepsilon(\hat{\boldsymbol{\nu}})}{\partial \hat{\boldsymbol{\nu}}}\right]^T \approx \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \left(\begin{bmatrix} J_x^2 & J_x J_y \\ J_x J_y & J_y^2 \end{bmatrix} \hat{\boldsymbol{\nu}} - \begin{bmatrix} \delta I J_x \\ \delta I J_y \end{bmatrix} \right). \tag{3.12}$$

Once computed the matrix

$$\mathbf{G} := \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} J_x^2 & J_x J_y \\ J_x J_y & J_y^2 \end{bmatrix} \tag{3.13}$$

and the vector

$$\mathbf{b} := \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} \delta I J_x \\ \delta I J_y \end{bmatrix} \tag{3.14}$$

the problem becomes

$$\frac{1}{2}\left[\frac{\partial\varepsilon(\hat{\boldsymbol{\nu}})}{\partial\hat{\boldsymbol{\nu}}}\right]^T \approx \mathbf{G}\hat{\boldsymbol{\nu}} - \mathbf{b} \tag{3.15}$$

that has solution: $\hat{\boldsymbol{\nu}}_{opt} = \mathbf{G}^{-1}\mathbf{b}$ This implementation is valid only for small pixel displacement, because of the 1st order Taylor approximation. In general a iterative version is needed.

3.4.2 Iterative implementation

The algorithm pseudo-code is taken integrally from [32] and reported below.

Goal: Let $\mathbf{u}$ be a point on image I. Find its corresponding location $\mathbf{v}$ on image J.
Build pyramid representation of I and J: $\{I^L\}_{L=0,\dots,L_m}$ and $\{J^L\}_{L=0,\dots,L_m}$
Initialization of pyramid guess: $g^{L_m} = [g_x^{L_m}\ g_y^{L_m}]^T = [0\ 0]^T$
for L=Lm down to 0 with step of -1
Location of point $\mathbf{u}$ on image I^L: $\mathbf{u}^L = [p_x\ p_y]^T = \frac{\mathbf{u}}{2^L}$
Derivative of J^L with respect to x: $J_x(x,y) = \dfrac{J^L(x+1,y) - J^L(x-1,y)}{2}$
Derivative of J^L with respect to y: $J_y(x,y) = \dfrac{J^L(x,y+1) - J^L(x,y-1)}{2}$
Spatial gradient matrix: $\mathbf{G} = \sum_{x=p_x-\omega_x}^{p_x+\omega_x}\sum_{y=p_y-\omega_y}^{p_y+\omega_y}\begin{bmatrix} J_x^2 & J_xJ_y \\ J_xJ_y & J_y^2 \end{bmatrix}$
Initialization of iterative LKT $\hat{\boldsymbol{\nu}}^0 = [0\ 0]^T$
for k=1 to K with step of 1 (or until $\|\hat{\boldsymbol{\eta}}^k\| <$ accuracy threshold)
Image difference: $\delta I_k(x,y) = I^L(x,y) - J^L(x + g_x^L + \hat{\nu}_x^{k-1}, y + g_y^L + \hat{\nu}_y^{k-1})$
Image mismatch vector: $\mathbf{b}_k = \sum_{x=p_x-\omega_x}^{p_x+\omega_x}\sum_{y=p_y-\omega_y}^{p_y+\omega_y}\begin{bmatrix} \delta IJ_x \\ \delta IJ_y \end{bmatrix}$
Optical flow: $\hat{\boldsymbol{\eta}}^k = \mathbf{G}^{-1}\mathbf{b}_k$
Guess for next iteration: $\hat{\boldsymbol{\nu}}^k = \hat{\boldsymbol{\nu}}^{k-1} + \hat{\boldsymbol{\eta}}^k$
end of for-loop on k
Final optical flow at level L: $\mathbf{d}^L = \hat{\nu}^K$
Guess for next level $L-1$: $\mathbf{g}^{L-1} = [g_x^{L-1}\ g_y^{L-1}]^T = 2(\mathbf{g}^L + \mathbf{d}^L)$
end of for-loop on L
Final optical flow vector: $\mathbf{d} = \mathbf{g}^0 + \mathbf{d}^0$
Location of point on J: $\mathbf{v} = \mathbf{u} + \mathbf{d}$
Solution: The corresponding point is at location $\mathbf{v}$ on image J.

3.5 BRISK

As explained in [33], the Binary Invariant Scalable Keypoints framework is used fo a high-quality, fast keypoint detection, description and matching. Its modularity allow to use the BRISK detector in combination with any other keypoint descriptor and vice versa. Only the only keypoint descriptor and matching part is described, since it is the part used in the OKVIS framework (4.4).

3.5.1 Keypoint descriptor

The BRISK descriptor is a binary string, composed of the results of brightness comparison tests concatenated together. The feature consist of sub-pixel refined image locations and associated floating-point scale values. The characteristic direction of each feature is identified to achieve rotation invariance. The brightness comparisons are selected with the focus on maximizing the image description.

Sampling pattern and rotation estimation

The BRISK descriptor uses a pattern for sampling the neighborhood of a keypoint. The patter, shown in figure 3.4, defines N points equally spaced on circles concentric with the keypoint. When sampling the intensity of a image point $\mathbf{p}_i$, to avoid aliasing effects, a Gaussian window is applied. The smoothing has a standard deviation σ_i, proportional to the distance between the points on the respective circle. Considering one of the $N(N-1)/2$ sampling-point pairs, $(\mathbf{p}_i, \mathbf{p}_j)$ and the respective smoothed intensity values, $I(\mathbf{p}_i, \sigma_i)$ and $I(\mathbf{p}_j, \sigma_j)$, the local gradient is:

$$\mathbf{g}(\mathbf{p}_i, \mathbf{p}_j) = (\mathbf{p}_j - \mathbf{p}_i)\frac{I(\mathbf{p}_j, \sigma_j) - I(\mathbf{p}_i, \sigma_i)}{\|\mathbf{p}_j - \mathbf{p}_i\|^2}. \tag{3.16}$$

Considering the set $\mathcal{A}$ of all sampling-point pairs:

$$\mathcal{A} = \{(\mathbf{p}_i, \mathbf{p}_j) \in \mathbb{R}^2 \times \mathbb{R}^2 \mid i < N \wedge j < i \wedge i, j \in \mathbb{N}\}, \tag{3.17}$$

a subset of short-distance pairings $\mathcal{S}$ and another subset of long-distance pairings $\mathcal{L}$ are defined:

$$\begin{aligned} \mathcal{S} &= \{(\mathbf{p}_i, \mathbf{p}_j) \in \mathcal{A} \mid \|\mathbf{p}_j - \mathbf{p}_i\| < \delta_{max}\} \subseteq \mathcal{A}, \\ \mathcal{L} &= \{(\mathbf{p}_i, \mathbf{p}_j) \in \mathcal{A} \mid \|\mathbf{p}_j - \mathbf{p}_i\| > \delta_{min}\} \subseteq \mathcal{A}. \end{aligned} \tag{3.18}$$

The threshold distances are set to $\delta_{max} = 9.75t$ and $\delta_{min} = 13.67t$ with t the scale of the keypoint k, since the algorithm can work on pyramidal patches. The long-distance pairs are used to estimate the overall characteristic pattern direction of the keypoint:

$$\mathbf{g} = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \frac{1}{L} \sum_{(\mathbf{p}_i, \mathbf{p}_j) \in \mathcal{L}} \mathbf{g}(\mathbf{p}_i, \mathbf{p}_j). \tag{3.19}$$

This is based on the assumption that local gradients cancel out each other.

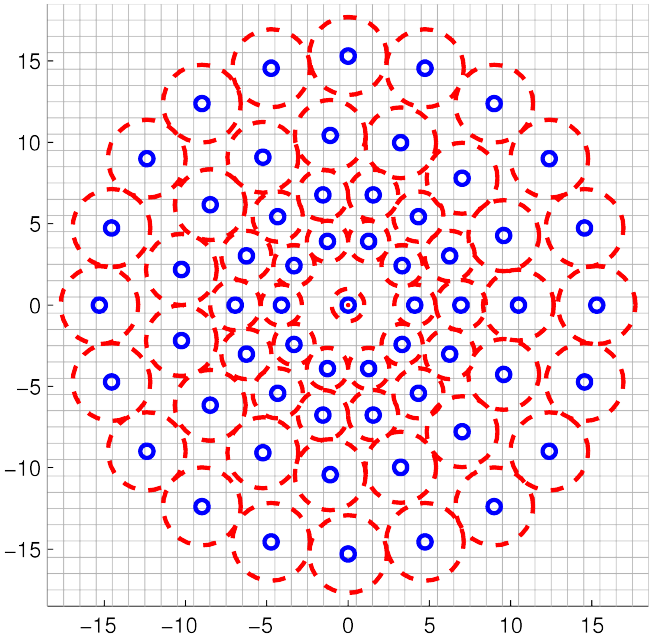


FIGURE 3.4: BRISK sampling pattern with $N = 60$. The mall blue circles denote the sampling locations and the bigger red dashed circles are drawn at a radius σ, corresponding to the standard deviation of the Gaussian kernel used to smooth the intensity values at the sampling points. The scale t os set to 1 [33].

Descriptor building

To create a rotation and scale invariant descriptor, BRISK applies the sampling pattern rotated by $\alpha = \arctan2(g_y, g_x)$ around the feature. The descriptor d_k is generate by performing the short distance intensity comparisons of all point pairs in the rotated pattern $(\mathbf{p}_i^\alpha, \mathbf{p}_j^\alpha) \in \mathcal{S}$, such that each bit b corresponds to:

$$\begin{cases} 1 & \text{if } I(\mathbf{p}_j^\alpha, \sigma_j) > I(\mathbf{p}_i^\alpha, \sigma_i) \\ 0 & \text{otherwise} \end{cases} \quad \forall (\mathbf{p}_i, \mathbf{p}_j) \in \mathcal{S}. \tag{3.20}$$

3.5.2 Descriptor matching

It is done by computing the descriptors Hamming distance: the number of bits different in the two descriptors is a measure of their dissimilarity. This reduces to a XOR operation followed by a bit count, which can both be computed very efficiently on today's architectures.

Chapter 4

Stereo visual inertial odometry algorithms

VIO algorithms combine visual odometry information coming from a stereo sensor with high frequency inertial data, to improve the estimation, especially making the system robust to fast motions.

4.1 Notation

Since this chapter presents a lot of mathematics, it is important to clarify the notation once and for all. All vectors are considered to be column vectors and written as boldface lowercase letters and the matrices are written as boldface uppercase letters. $\mathbf{I}$ is the identity matrix, with dimensions in agreement with the equation it is in. Three different coordinate frames are used in this exposition: the inertial world coordinate frame, $\mathcal{W}$, the IMU fixed coordinate frame, $\mathcal{B}$, as well as the camera fixed coordinate frame, $\mathcal{C}$. The origin associated with a coordinate frame is referred as by the same symbol. For example ${}_{\mathcal{W}}\mathbf{r}_{\mathcal{C}}^{\mathcal{B}}$ identifies the coordinates of a vector from the origin of $\mathcal{B}$ to the origin of $\mathcal{C}$, expressed in the coordinate frame $\mathcal{W}$. To simplify the notation, it is used this equivalence ${}_{\mathcal{B}}\mathbf{r}_{\mathcal{C}}^{\mathcal{B}} := \mathbf{r}_{\mathcal{C}}^{\mathcal{B}}$ For the rotations, $\mathbf{u}_{\mathcal{W}}^{\mathcal{B}}$ represents the rotation between a frame $\mathcal{W}$ and $\mathcal{B}$ and it is defined as a mapping $\mathbf{u}_{\mathcal{W}}^{\mathcal{B}} : \mathbb{R}^3 \to \mathbb{R}^3$. The vector $\mathbf{u}_{\mathcal{W}}^{\mathcal{B}}$ defines the change to a rotated reference frame, such that ${}_{\mathcal{B}}\mathbf{r}_{\mathcal{C}}^{\mathcal{B}} = \mathbf{u}_{\mathcal{W}}^{\mathcal{B}}({}_{\mathcal{W}}\mathbf{r}_{\mathcal{C}}^{\mathcal{B}})$. In the following part it is going to be used also another mapping $\mathbf{C}(\mathbf{u}) : SO(3) \to \mathbb{R}^{3\times3}$, where SO(3) is the 3D rotation group and which is defined such that $\mathbf{u}(\mathbf{r}) \triangleq \mathbf{C}(\mathbf{u})\mathbf{r}$ and essentially returns the rotation matrix. Moreover, $\mathbf{v}_{\mathcal{B}}$ represents the absolute velocity of $\mathcal{B}$, and $\boldsymbol{\omega}_{\mathcal{W}}^{\mathcal{B}}$ for the vector describing the relative rotational velocity of the coordinate frame $\mathcal{B}$ with respect to the coordinate frame $\mathcal{W}$. The superscript $^{\times}$ is used to denote the skew symmetric matrix $\mathbf{v}^{\times} \in \mathbb{R}^{3\times3}$ of a vector $\mathbf{v} \in \mathbb{R}^3$. Defining $\mathbf{v} = [v_x \; v_y \; v_z]^T$, the skew symmetric matrix becomes:

$$\mathbf{v}^{\times} = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix} \tag{4.1}$$

Moreover a quaternion is defined as $\mathbf{q} = [q_w \; q_x \; q_y \; q_z]^T$ with real part $q_w \in \mathbb{R}$ and imaginary part $\mathbf{q}_v = [q_x \; q_y \; q_z]^T \in \mathbb{R}^3$. $[\mathbf{q}]_{xyz}$, extracts the vector part of a quaternion $\mathbf{q}$. $\hat{(\cdot)}$ identifies the estimated value and $\tilde{(\cdot)}$ the measurement.

4.2 ROVIO

All the material that is not explicitly referenced in this section 4.1 about ROVIO comes from [34] and [35].
Robust Visual Inertial Odometry is a visual-inertial odometry framework with the following main characteristics:

- It is used an iterative extended Kalman filter that inherently performs per-landmark tracking and online camera-IMU extrinsics calibration,
- It is used a robocentric formulation of the full state and manifold encapsulation for rotations and feature bearing vectors,
- It is applied a FAST corner detector and at each feature it are associated multi-level patches, which intensity error is used in the innovation step,
- QR decomposition is used to obtain a 2D innovation term for every observed landmark, keeping the update computation tractable.

4.2.1 Representation of 3D rotations

The set of 3D rotations, defined as the special orthogonal group SO(3) with group operation $\otimes$, is not a vector space and to use optimization methods some adjustments are needed. The common solutions consist in [36] :

- Using a minimal dimension parametrization of 3 parameters as the Euler angles and operate on them as in the Euclidean space. The main drawback is the presence of singularities that consist in situation in which very large changes in the parameters are needed to represent small variations in the state space,
- Adopting a non-minimal representation as rotation matrices ($\mathbb{R}^{3\times3}$) and unit quaternions ($\mathbb{R}^4$). A re-normalization is needed to avoid degenerate parameter. This solution is not optimal from a memory efficiency point of view, since it introduces redundant values.

A more general solution is proposed in [35] follows. The region around a selected linearization point is mapped to a proper vector space and a local parametrization is used. SO(3) is a Lie group and has a logarithmic and an exponential map which map to

and from a corresponding Lie algebra $\mathbb{R}^3$. The map is selected such that $\boldsymbol{\theta}_{\mathcal{W}}^{\mathcal{B}}$ coincides with the passive rotation vector of the rotation $\mathbf{u}_{\mathcal{W}}^{\mathcal{B}}$.

$$\begin{aligned} &log : SO(3) \to \mathbb{R}^3 \\ &\mathbf{u}_{\mathcal{W}}^{\mathcal{B}} \mapsto log(\mathbf{u}_{\mathcal{W}}^{\mathcal{B}}) = \boldsymbol{\theta}_{\mathcal{W}}^{\mathcal{B}} \end{aligned} \tag{4.2}$$

$$\begin{aligned} &exp : \mathbb{R}^3 \to SO(3) \\ &\boldsymbol{\theta}_{\mathcal{W}}^{\mathcal{B}} \mapsto exp(\boldsymbol{\theta}_{\mathcal{W}}^{\mathcal{B}}) = \mathbf{u}_{\mathcal{W}}^{\mathcal{B}} \end{aligned} \tag{4.3}$$

The following identities are verified by these maps:

$$exp(-\boldsymbol{\theta}) = exp(-\boldsymbol{\theta})^{-1} \tag{4.4}$$

$$exp(\mathbf{u}(\boldsymbol{\theta})) = \mathbf{u} \otimes exp(\boldsymbol{\theta}) \otimes \mathbf{u}^{-1} \tag{4.5}$$

$$\mathbf{C}(\boldsymbol{\theta}) = \mathbf{I} - \frac{\sin(\|\boldsymbol{\theta}\|)\boldsymbol{\theta}^{\times}}{\|\boldsymbol{\theta}\|} + \frac{1 - \cos(\|\boldsymbol{\theta}\|)\boldsymbol{\theta}^{\times^2}}{\|\boldsymbol{\theta}\|^2} \tag{4.6}$$

The maps can be used to introduce a boxplus ($\boxplus$) and a boxminus ($\boxminus$) operator, which adopt the role of addition and subtraction operators for rotations. They are defined as:

$$\begin{aligned} &\boxplus : SO(3) \times \mathbb{R}^3 \to \mathbb{R}^3 \\ &\mathbf{u}, \boldsymbol{\theta} \mapsto exp(\boldsymbol{\theta}) \otimes \mathbf{u} \end{aligned} \tag{4.7}$$

$$\begin{aligned} &\boxminus : SO(3) \times SO(3) \to \mathbb{R}^3 \\ &\mathbf{u}, \mathbf{s} \mapsto log(\mathbf{u} \otimes \mathbf{s}^{-1}) \end{aligned} \tag{4.8}$$

Both operators fulfill the following axioms, similar to regular addition and subtraction:

$$\mathbf{u} \boxplus \mathbf{0} = \mathbf{u} \tag{4.9}$$

$$(\mathbf{u} \boxplus \boldsymbol{\theta}) \boxminus \mathbf{u} = \boldsymbol{\theta} \tag{4.10}$$

$$\mathbf{u} \boxplus (\mathbf{s} \boxminus \mathbf{u}) = \mathbf{s} \tag{4.11}$$

Using this approach there is a distinction between actual rotation, which are on the Lie group SO(3) and difference of rotation, which belong to the Lie algebra $\mathbb{R}^3$. Using the boxplus and boxminus instead of the regular operators allows to define differentials involving rotation. In this way can be obtained derivatives that do not depend on a specific parametrization (e.g. quaternions or rotation matrices), but only if the parametrization is lossless and the associated operations follow certain rules. For instance, the differential of a mapping $\mathbf{u}(x) : \mathbb{R} \to SO(3)$ is defined as:

$$\frac{\partial}{\partial x}\mathbf{u}(x) := \lim_{\varepsilon \to 0} \frac{\mathbf{u}(x + \varepsilon) \boxminus \mathbf{t}(\mathbf{x})}{\varepsilon}. \tag{4.12}$$

The same can be done for the other way round, with a mapping $x(\mathbf{u}) : SO(3) \to \mathbb{R}$:

$$\frac{\partial}{\partial \mathbf{u}} x(\mathbf{u}) := \lim_{\varepsilon \to 0} \begin{bmatrix} \frac{x(\mathbf{u} \boxplus (\varepsilon \mathbf{e}_1)) - x(\mathbf{u})}{\varepsilon} \\ \frac{x(\mathbf{u} \boxplus (\varepsilon \mathbf{e}_2)) - x(\mathbf{u})}{\varepsilon} \\ \frac{x(\mathbf{u} \boxplus (\varepsilon \mathbf{e}_3)) - x(\mathbf{u})}{\varepsilon} \end{bmatrix}, \tag{4.13}$$

where $\mathbf{e}_{1/2/3}$ are orthonormal basis vectors. This results in the following derivatives:

$$\begin{aligned} &\frac{\partial}{\partial t} \mathbf{u}_{\mathcal{W}}^{\mathcal{B}}(t) = \boldsymbol{\omega}_{\mathcal{W}}^{\mathcal{B}}(t), \\ &\frac{\partial}{\partial \mathbf{u}} \mathbf{u}(\mathbf{r}) = \mathbf{u}(\mathbf{r})^{\times}, \\ &\frac{\partial}{\partial \mathbf{u}} \mathbf{u}^{-1} = -\mathbf{C}(\mathbf{u})^{T}, \\ &\frac{\partial}{\partial \mathbf{u}} (\mathbf{u} \otimes \mathbf{s}) = \mathbf{I}, \\ &\frac{\partial}{\partial \mathbf{u}} (\mathbf{s} \otimes \mathbf{u}) = \mathbf{C}(\mathbf{s}), \\ &\frac{\partial}{\partial \boldsymbol{\theta}} (\exp(\boldsymbol{\theta})) = \boldsymbol{\Gamma}(\boldsymbol{\theta}), \\ &\frac{\partial}{\partial \mathbf{u}} (\log(\mathbf{u})) = \boldsymbol{\Gamma}^{-1}(\log(\mathbf{u})). \end{aligned} \tag{4.14}$$

The exponential map derivative is given by the Jacobian $\boldsymbol{\Gamma} \in \mathbb{R}^{3\times3}$ obtained from:

$$\boldsymbol{\Gamma}(\boldsymbol{\theta}) = \mathbf{I} - \frac{(1 - \cos(\|\boldsymbol{\theta}\|))\boldsymbol{\theta}^{\times}}{\|\boldsymbol{\theta}\|^2} + \frac{(\|\boldsymbol{\theta}\| - \sin(\|\boldsymbol{\theta}\|))\boldsymbol{\theta}^{\times^2}}{\|\boldsymbol{\theta}\|^3} \tag{4.15}$$

The parametrization chosen is based on unit quaternions using Hemilton convention. Considering a unit quaternion $\mathbf{q}$, with real part q_w and imaginary part $\mathbf{q}_v$, the following maps are employed:

$$exp(\boldsymbol{\theta}) = [q_w, \mathbf{q}_v] = \left[\cos\left(\frac{\|\boldsymbol{\theta}\|}{2}\right), \sin\left(\frac{\|\boldsymbol{\theta}\|}{2}\right) \frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|} \right] \tag{4.16}$$

$$log(\mathbf{q}) = 2\arctan2\left(\|\mathbf{q}_v\|, q_w\right) \frac{q_w}{\|\mathbf{q}_v\|} \tag{4.17}$$

Both maps and the quaternion multiplication, are the only mathematical operations required that are bound to the parametrization.

4.2.2 Representation of 3D unit vectors

The authors of [35] extend the use of local mapping also to 3D unit vectors on the 2-sphere S^2 (a sphere in the Euclidean 3D space). It is employed a parametrization that leads to simple analytical derivatives and guarantees second order differentiability. There are two problems:

- There is no continuous way to assign orthonormal vectors to every point of a 2-sphere
- Orthonormal vectors have to be selected, to span the tangent space such that a suitable difference operator can be defined

In over to overcome these issues, it is defined a rotation, $\boldsymbol{\mu} \in SO(3)$ and the following quantities:

$$\begin{aligned} &\mathbf{n}(\boldsymbol{\mu}) := \boldsymbol{\mu}(\mathbf{e}_z) \in S^2 \subset \mathbb{R}^3, \\ &\mathbf{N}(\boldsymbol{\mu}) := [\boldsymbol{\mu}(\mathbf{e}_x), \boldsymbol{\mu}(\mathbf{e}_y)] \in S^2 \subset \mathbb{R}^{3\times 2}, \end{aligned} \tag{4.18}$$

where $\mathbf{e}_{x/y/z} \in \mathbb{R}^3$ are the basis vectors of a orthonormal coordinate system. $\mathbf{n}(\boldsymbol{\mu})$ is the actual unit vector resulting when $\mathbf{e}_z$ is rotated by $\boldsymbol{\mu}$. The matrix $\mathbf{N}(\boldsymbol{\mu})$ is composed of the rotated $\mathbf{e}_x$ and $\mathbf{e}_y$ and covers the tangent space. The tangent space can be used to define the following boxplus and boxminus operators:

$$\begin{aligned} &\boxplus : SO(3) \times \mathbb{R}^2 \to SO(3), \\ &\boldsymbol{\mu}, \mathbf{u} \mapsto exp(\mathbf{N}(\boldsymbol{\mu})\mathbf{u}) \otimes \boldsymbol{\mu}, \end{aligned} \tag{4.19}$$

$$\begin{aligned} &\boxminus : SO(3) \times SO(3) \to \mathbb{R}^2, \\ &\boldsymbol{\nu}, \boldsymbol{\mu} \mapsto \mathbf{N}(\boldsymbol{\mu})^T \boldsymbol{\theta}(\boldsymbol{\mu}, \boldsymbol{\nu}), \end{aligned} \tag{4.20}$$

where $\boldsymbol{\theta}$ maps two unit vectors to the minimal rotation vector between them:

$$\boldsymbol{\theta}(\mathbf{n}(\boldsymbol{\mu}), \mathbf{n}(\boldsymbol{\nu})) = \frac{\arccos(\mathbf{n}(\boldsymbol{\nu})^T \mathbf{n}(\boldsymbol{\mu})}{\|\mathbf{n}(\boldsymbol{\nu}) \times \mathbf{n}(\boldsymbol{\mu})\|} \mathbf{n}(\boldsymbol{\nu}) \times \mathbf{n}(\boldsymbol{\mu}). \tag{4.21}$$

To better understand what has been presented, a visualization of the 2-sphere and the tangent space for a specific $\boldsymbol{\mu}$ is given in fig. 4.1. To overcome the fact that the tangent space is not deterministic, it is defined a notion of equivalence as that two unit vector parametrizations $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ are equivalent ($\boldsymbol{\mu} \sim \boldsymbol{\nu}$) *iff* $\mathbf{n}(\boldsymbol{\mu}) = \mathbf{n}(\boldsymbol{\nu})$. Using this definition, the following axioms are valid:

$$\begin{aligned} &\boldsymbol{\mu} \boxplus \mathbf{0} = \boldsymbol{\mu}, \\ &(\boldsymbol{\mu} \boxplus \mathbf{u}) \boxminus \boldsymbol{\mu} = \mathbf{u}, \\ &\boldsymbol{\mu} \boxplus (\boldsymbol{\nu} \boxminus \boldsymbol{\mu}) \sim \boldsymbol{\nu}. \end{aligned} \tag{4.22}$$

It is important to point out that whenever representing a difference, $\mathbf{u} \in \mathbb{R}^2$, the corresponding tangent space is important. In fact, considering two rotations $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ with $\boldsymbol{\mu} \sim \boldsymbol{\nu}$, it does not result that $(\boldsymbol{\mu} \boxplus \mathbf{u}) \sim (\boldsymbol{\nu} \boxplus \mathbf{u})$. From this parametrization it is straightforward to obtain the derivatives needed when computing Jacobians. The

most commonly used ones are:

$$\begin{aligned}
&\frac{\partial}{\partial t}\boldsymbol{\mu}(t) = -\mathbf{N}(\boldsymbol{\mu}(t))^T\mathbf{n}(\boldsymbol{\mu}(t))^\times\frac{\partial}{\partial t}\mathbf{n}(\boldsymbol{\mu}(t)),\\
&\frac{\partial}{\partial \boldsymbol{\mu}}\mathbf{n}(\boldsymbol{\mu}) = \mathbf{n}(\boldsymbol{\mu})^\times\mathbf{N}(\boldsymbol{\mu}),\\
&\frac{\partial}{\partial \boldsymbol{\mu}}(\mathbf{N}(\boldsymbol{\mu})^T\mathbf{r}) = -\mathbf{N}(\boldsymbol{\mu})^T\mathbf{r}^\times\mathbf{N}(\boldsymbol{\mu}).
\end{aligned} \tag{4.23}$$

This parametrization has multiple advantages:

- It is singualrity-free and with relatively simple differentials
- The $\boxminus$ operation always gives the minimum rotation, since the tangent space has an orthogonal particularization and the boxminus operation has a direction which spans the shortest path between two unit vectors.

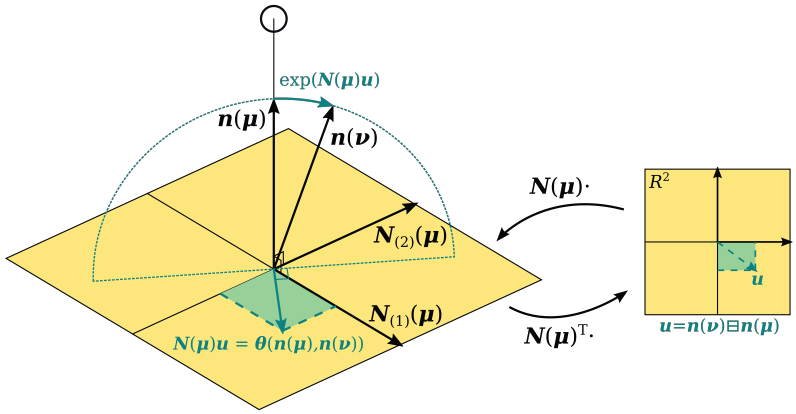
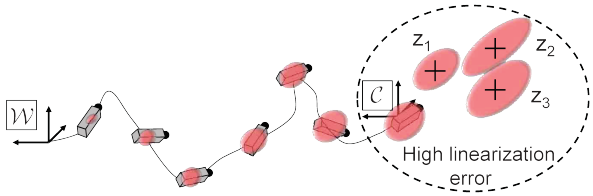


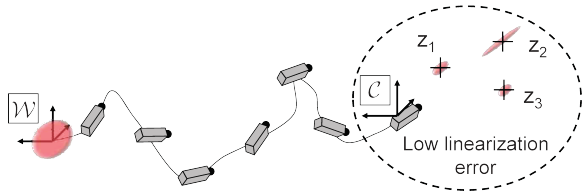
FIGURE 4.1: Representation of 3D unit vectors. The 3D unit vector $\mathbf{n}(\boldsymbol{\mu})$ is considered the result of a rotation $\boldsymbol{\mu}$ applied to the z axis of a reference frame. The other axes define a orthonormal plane to the vector. [35]

4.2.3 Robocentric state representation

As stated in [37], it refers to the practice of representing the geometric state variables in a reference frame rigid with the camera. This results in a low uncertainty in the vicinity of the sensor during the entirety of the run, thus in reduction of the linearization errors associated with the measurement model. For example, considering an VO framework based on an Extended Kalman Filter (EKF), as shown in fig. 4.2a, as the camera moves further from the world reference frame, non revisiting the known places k, both the camera uncertainty and features uncertainty grow. Whereas, using a camera-centered EKF estimation (fig. 4.2b), the camera location uncertainty is near zero, since the reference frame is attached to it.



(A) World centered state representation



(B) Camera centered state representation

FIGURE 4.2: World centered vs camera centered state representation. z_* are the features in the camera proximity and the red areas show the position uncertainty [37].

4.2.4 Projection model and linear warping

Landmarks are modeled as fixed locations in the world with a multilevel patch (sec. 3.1.3) associated to each one of them. The image levels have a scale factor of 2. The knowledge of the intrinsic camera model can be leveraged to project features from an image to another one. This is important because, when the camera moves also the feature do, changing the bearing. Assuming a known intrinsic camera model $\boldsymbol{\pi}_c$ and given the bearing vector $\mathbf{n}(\boldsymbol{\mu})$ of a landmark, the pixel coordinates p can be expressed as $p = \boldsymbol{\pi}_c(\mathbf{n}(\boldsymbol{\mu}))$. The camera movement is described by a warping matrix for each feature and it is mathematically defined by a concatenation of Jacobians. For instance, if it is detected a feature in some frame at pixel p_1, with the correspondent bearing vector $\mathbf{n}(\boldsymbol{\mu}_1) = \boldsymbol{\pi}_c{}^{-1}(\mathbf{p}_1)$. The latter is transformed by process model $\mathbf{n}(\boldsymbol{\mu}_2) = \mathbf{f}(\mathbf{n}(\boldsymbol{\mu}_1))$ and then re-projected in the following frame $\mathbf{p}_2 = \boldsymbol{\pi}_c(\mathbf{n}(\boldsymbol{\mu}_2))$. This consists in the warping matrix:

$$\mathbf{D} = \frac{\partial \boldsymbol{\pi}_c(\mathbf{n}(\boldsymbol{\mu_2}))}{\partial \mathbf{n}(\boldsymbol{\mu_2})} \frac{\partial \mathbf{f}(\mathbf{n}(\boldsymbol{\mu_1}))}{\partial \mathbf{n}(\boldsymbol{\mu_1})} \frac{\partial \boldsymbol{\pi}_c^{-1}(\mathbf{p}_1)}{\partial \mathbf{p}_1} \in \mathbb{R}^{2\times 2} \tag{4.24}$$

This map is used to warp patches in new frames. It is just an approximation since only the local distortion is considered, ignoring larger scale information as the patch shape. To avoid highly distorted patches, they are re-extracted regularly and the warping matrix is reset to identity.

4.2.5 Photometric error and patch alignment

ROVIO implies a patch alignment algorithm similar to the KLT feature tracker (3.4). Similarly to that method, the photometric error is defined between the intensity difference of pixels intensity between two images, but in this case is the warped patch

is considered. Moreover, in this case the error is defined as a stack or errors for each image level. Considering a multilevel patch feature with coordinates p and multilevel patch $P = (P_0, ..., P_L)$. The photometric error of a patch pixel i at level l is written as:

$$e_{l,i}(\mathbf{p}, P, I, \mathbf{D}) = P_l(\mathbf{p}_i) - aI_l(\mathbf{p}s_l + \mathbf{D}\mathbf{p}_i) - b \tag{4.25}$$

where I_l is the image at the pyramid level l and $s_l = 0.5^l$ is the scaling factor between levels. $\mathbf{D}$ is the linear warping matrix. The parameters a and b are part of an affine intensity model (an affine model is based on an affine function, that is the composition of a linear function with a translation parameter, e.g. $f(x) = ax + b$ [38]) to account for illumination changes between frames.
The patch alignment is preformed minimizing the squared error terms. One way to solve the minimization is with a Gauss-Newton method that applies a local approximation of the problem, linearizing around an estimated patch location p:

$$\mathbf{b}(\mathbf{p} + \delta\mathbf{p}, P, I, \mathbf{D}) = \mathbf{A}(\mathbf{p}, I, \mathbf{D})\delta\mathbf{p} + \mathbf{b}(\mathbf{p}, P, I, \mathbf{D}) \tag{4.26}$$

where $\mathbf{b}(\mathbf{p}, P, I, \mathbf{D})$ represents the error terms from eq. 4.25 summed over the patch pixels and stacked together, while $\mathbf{A}(\mathbf{p}, I, \mathbf{D})$ is the corresponding Jacobian. Applying the first order Taylor expansion the problem becomes:

$$\mathbf{A}(\mathbf{p}, I, \mathbf{D})^T\mathbf{A}(\mathbf{p}, I, \mathbf{D})\delta\mathbf{p} = -\mathbf{A}(\mathbf{p}, I, \mathbf{D})^T\mathbf{b}(\mathbf{p}, P, I, \mathbf{D}) \tag{4.27}$$

which can be solved for, $\delta\mathbf{p}$. This is analogous to one iteration step of the KLT feature tracker. The scaling factor s_l in eq. 4.25 causes the error terms for higher image levels to have a minimum influence in the patch alignment, which is consistent with the fact that they carry less information. However the multiple levels are important to increase the method robustness in case of image blur and bad initial alignment.
The photometric error inherently relies on past information in the shape of a patch from a previous image and this causes cross-correlations between the error and the states. This arises when the information from parts of a patch has influenced the past filter states. This patch has then a pixel intensity error correlated with the current state. This is not modeled to avoid an excessive increase of the computation requirements.

4.2.6 Detection and scoring

The FAST detector (3.2) is used to produce a large number of possible feature candidates. First the feature candidates are spaced by enforcing a minimum distance from the ones that are already tracked, then an adaptation of the Shi-Tomasi score (3.3.2) is applied to find the new features to track. The score applied considers the Hessian matrix of multiple image levels, instead of only a single level. The combined Hessian can be found from the eq. 4.27:

$$\mathbf{H} = \mathbf{A}(\mathbf{p}, I, \mathbf{D})^T\mathbf{A}(\mathbf{p}, I, \mathbf{D})\delta\mathbf{p} \tag{4.28}$$

where the minimal eigenvalue of $\mathbf{H}$ is just the Shi-Tomasi score (sec. 3.3.2) modified for the multiple levels.
In environments with limited corner features other scores based ot the Hessian eigenvalues are enforced, increasing the feature count with edge features. Another bucketing technique (that consists in dividing the image space in parts and tracking only a number of features for each part) is applied, to have a good coverage of the image.

4.2.7 Iterated extended Kalman filtering

The extended Kalman filter is widely used in data fusion frameworks. It represents an adaptation of the Kalman filter for the non-linear case, using a first order Taylor series to linear the model with Jacobian matrices that need to be recomputed each iteration [39]. This does not lead to the optimal solution for a non-linear system, but in many cases it keeps it close to the optimum and maintains the problem computationally tractable.
One way to improve the filter accuracy is to make use of an iterative form to refine the a-posteriori estimate. A discrete time non-linear system can be formalized as:

$$\begin{aligned} \mathbf{x}_k &= \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{n}_{k-1}), \\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k, \mathbf{n}_k), \end{aligned} \tag{4.29}$$

where x is the state, y the innovation term, $\mathbf{w} \sim \mathcal{N}(0, \mathbf{W})$ is the process noise and $\mathbf{n} \sim \mathcal{N}(0, \mathbf{R})$ the update noise.
The IEKF prediction step is defined as:

$$\begin{aligned} \mathbf{x}_k^- &= \mathbf{f}(\mathbf{x}_{k-1}^+, \mathbf{0}), \\ \mathbf{P}_k^- &= \mathbf{F}_{k-1}\mathbf{P}_{k-1}^+\mathbf{F}_{k-1}^T + \mathbf{G}_{k-1}\mathbf{W}_{k-1}\mathbf{G}_{k-1}^T, \end{aligned} \tag{4.30}$$

where $\mathbf{x}_{k-1}^+$ is the a-posteriori estimate from the previous iteration, with covariance $\mathbf{P}_{k-1}^+$ and $\mathbf{x}_k^-$ is the new a-priori estimate.
The Jacobians are:

$$\begin{aligned} \mathbf{F}_{k-1} &= \frac{\partial \mathbf{f}}{\partial \mathbf{x}_{k-1}}(\mathbf{x}_{k-1}^+, \mathbf{0}), \\ \mathbf{G}_{k-1} &= \frac{\partial \mathbf{f}}{\partial \mathbf{w}_{k-1}}(\mathbf{x}_{k-1}^+, \mathbf{0}). \end{aligned} \tag{4.31}$$

The IEKF update step, as the EKF one, can be expressed as an optimization problem:

$$\min_{\mathbf{x}_k^+} \|\mathbf{x}_k^+ \boxminus \mathbf{x}_k^-\|_{\mathbf{P}_k^{-1}} + \|\mathbf{h}(\mathbf{x}_k^+, \mathbf{0})\|_{(\mathbf{J}_k\mathbf{R}_k\mathbf{J}_k^T)^{-1}} \tag{4.32}$$

where it is reported the deviation from $\mathbf{x}_k^-$ and the innovation term $\mathbf{h}(\mathbf{x}_k^+, \mathbf{0})$.
However, in the Iterated Extended Kalman Filter (IEKF), is involved an iteration starting with $\mathbf{x}_{k,0}^+ = \mathbf{x}_k^-$ in which the linearization around continuously refined linearization

points $\mathbf{x}_{k,j}^{+}$. Therefore, the IEKF update step is better formalized as:

$$\min_{\Delta\mathbf{x}_{k,j}^{+}} \|\mathbf{x}_{k,j}^{+} \boxminus \mathbf{x}_{k}^{-} + \mathbf{L}_{k,j}^{-1}\Delta\mathbf{x}_{k,j}\|_{\mathbf{P}_{k}^{-}{}^{-1}} + \|\mathbf{h}(\mathbf{x}_{k}^{+}, \mathbf{0}) + \mathbf{H}_{k,j}\Delta\mathbf{x}_{k,j}\|_{(\mathbf{J}_k\mathbf{R}_k\mathbf{J}_k^T)^{-1}} \tag{4.33}$$

where the Jacobians are updated every iteration step:

$$\begin{aligned}
\mathbf{H}_{k,j} &= \frac{\partial \mathbf{h}}{\partial \mathbf{x}_k}(\mathbf{x}_{k,j}^{+}, \mathbf{0}), \\
\mathbf{J}_{k,j} &= \frac{\partial \mathbf{h}}{\partial \mathbf{n}_k}(\mathbf{x}_{k,j}^{+}, \mathbf{0}), \\
\mathbf{L}_{k,j} &= \frac{\partial (\mathbf{x}_{k}^{-} \boxplus \Delta\mathbf{x})}{\partial \Delta\mathbf{x}}(\mathbf{x}_{k,j}^{+} \boxminus \mathbf{x}_{k}^{-}).
\end{aligned} \tag{4.34}$$

The iterative representation of the optimization problem is obtained by setting the derivative of the cost function 4.33 with respect to the incremental update $\Delta\mathbf{x}_{k,j}$ to zero and applying some matrix calculus. it results in the following:

$$\begin{aligned}
&\mathbf{S}_{k,j} = \mathbf{H}_{k,j}\mathbf{L}_{k,j}^{T}\mathbf{P}_{k}^{-}\mathbf{L}_{k,j}\mathbf{H}_{k,j}^{T} + \mathbf{J}_{k,j}\mathbf{R}_k\mathbf{J}_{k,j}^{T}, \\
&\mathbf{K}_{k,j} = \mathbf{L}_{k,j}^{T}\mathbf{P}_{k}^{-}\mathbf{L}_{k,j}\mathbf{H}_{k,j}^{T}\mathbf{S}_{k,j}^{-1}, \\
&\Delta\mathbf{x}_{k,j} = \mathbf{K}_{k,j}\left(\mathbf{H}_{k,j}\mathbf{L}_{k,j}(\mathbf{x}_{k,j}^{+} \boxminus \mathbf{x}_{k}^{-}) - \mathbf{h}(\mathbf{x}_{k,j}^{+}, \mathbf{0})\right) - \mathbf{L}_{k,j}(\mathbf{x}_{k,j}^{+} \boxminus \mathbf{x}_{k}^{-}), \\
&\mathbf{x}_{k,j+1}^{+} = \mathbf{x}_{k,j}^{+} \boxplus \Delta\mathbf{x}_{k,j},
\end{aligned} \tag{4.35}$$

where the stopping criteria is $\Delta\mathbf{x}_{k,j} < T$, with T a threshold. Once the process has converged the estimate covariance matrix is updated:

$$\mathbf{P}_{k}^{+} = (\mathbf{I} - \mathbf{K}_{k,n}\mathbf{H}_{k,n})\mathbf{L}_{k,n}^{T}\mathbf{P}_{k}^{-}\mathbf{L}_{k,n} \tag{4.36}$$

with n the number of iterations. In situations with large initial uncertainties, as a landmark initialization, the IEKF can lead to an important improvement in convergence and accuracy. A termination criteria proportional to the performed correction can be used to limit the computation time. The filter performs really close to a regular EKF, when the state has converged to a mostly stable value, since only a minimum number of iteration is needed.

4.2.8 Filter setup and state definition

The filter framework (fig. 4.3), uses a robocentric state formulation to decouple the observable and unobservable states (position and yaw). This decreases the noise in the relevant states, improving their consistency, but also subjects them to the gyroscope noise due to the rotation in the robocentric frame. In the end is worth it, since only few states are unobservable and the gyroscope noise is usually small. The state is defined as:

$$\mathbf{x} := [\mathbf{r}\ \mathbf{v}\ \mathbf{q}\ \mathbf{b}_f\ \mathbf{b}_g\ \mathbf{c}\ \mathbf{z}\ \mathbf{n}(\boldsymbol{\mu}_0)\ ...\ \mathbf{n}(\boldsymbol{\mu}_N)\ \rho_0\ ...\ \rho_N] \tag{4.37}$$

with:

- $\mathbf{r} := \mathbf{r}_{\mathcal{W}}^{\mathcal{B}}$: robocentric position of IMU
- $\mathbf{v} := {}_{\mathcal{B}}\mathbf{v}_{\mathcal{B}}$: robocentric velocity of IMU
- $\mathbf{q} := \mathbf{q}_{\mathcal{W}}^{\mathcal{B}}$: attitude of IMU
- $\mathbf{b}_f$: additive bias on accelerometer (expressed in B)
- $\mathbf{b}_g$: additive bias on gyroscope (expressed in B)
- $\mathbf{c} := \mathbf{r}_{\mathcal{C}}^{\mathcal{B}}$: linear part of IMU-camera extrinsics
- $\mathbf{z} := \mathbf{q}_{\mathcal{C}}^{\mathcal{B}}$: rotational part of IMU-camera extrinsics
- $\mathbf{n}(\boldsymbol{\mu}_i)$: bearing vector to landmark i (expressed in C)
- ρ_i: distance parameter of landmark i

The distance d_i of a landmark i can be parameterized by different mappings $d(\rho_i)$ with derivative $d'(\rho_i)$. In the last implementation of Rovio the possible mappings are:

- Regular: $d(\rho_i) = \rho_i$,
- Inverse: $d(\rho_i) = \dfrac{1}{\rho_i}$,
- Logarithmic: $d(\rho_i) = \mathrm{e}^{\rho_i}$,
- Hyperbolic: $d(\rho_i) = \sinh(\rho_i)$.

Rotations $(\mathbf{q}, \mathbf{z})$ and bearing vectors $\mathbf{n}(\boldsymbol{\mu}_i)$ are expressed with a minimal and singularity-free parametrization, as shown in section 4.2.1 and section 4.2.2. This allows for the landmarks to be immediately initialized when detected and to minimze the computation and memory burden of the filter.

4.2.9 State propagation

For state propagation a IMU and camera movement model are used. The IMU model takes in account a random walk bias and a Gaussian additive noise:

$$\begin{aligned} \mathbf{a} &= \tilde{\mathbf{a}} - \mathbf{b}_a - \mathbf{n}_a, \\ \boldsymbol{\omega} &= \tilde{\boldsymbol{\omega}} - \mathbf{b}_g - \mathbf{n}_\omega, \end{aligned} \tag{4.38}$$

where $\tilde{\mathbf{a}} = {}_{\mathcal{B}}\tilde{\mathbf{a}}_{\mathcal{B}}$ is the acceleration measurement and $\tilde{\boldsymbol{\omega}} = {}_{\mathcal{B}}\tilde{\boldsymbol{\omega}}_{\mathcal{B}}$ is the rotational rate measurement. The camera model uses a constant velocity and consists in:

$$\begin{aligned} \mathbf{v}_{\mathcal{C}} &= \mathbf{z}(\mathbf{v} + \boldsymbol{\omega}^{\times}\mathbf{c}), \\ \boldsymbol{\omega}_{\mathcal{C}} &= \mathbf{z}(\boldsymbol{\omega}). \end{aligned} \tag{4.39}$$

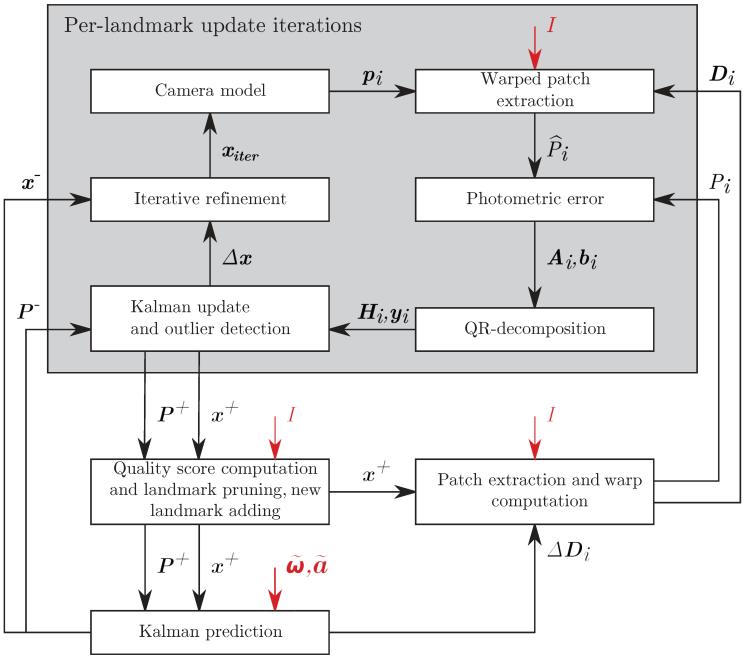


FIGURE 4.3: ROVIO framework diagram [35].

Moreover, the bearing vector and distance parameter act on a 3D point landmark following this equation:

$$\mathbf{r}_{\mathcal{F}}^{\mathcal{W}} = \mathbf{r}_{\mathcal{C}}^{\mathcal{W}} + \mathbf{q}_{\mathcal{W}}^{\mathcal{C}}{}^{-1}(\mathbf{n}(\boldsymbol{\mu})\ d(\rho)) \tag{4.40}$$

in which is taken in account a landmark with a reference frame $\mathcal{F}$ attached to it, so with position $\mathbf{r}_{\mathcal{F}}^{\mathcal{W}}$, bearing vector $\mathbf{n}(\boldsymbol{\mu})$ and distance parameter ρ. This landmark is assumed to be static with respect to the world reference frame.
The time differentiation of these models leads to the following system of continuous differential equations (the full derivation is reported in the original paper):

$$\dot{\mathbf{r}} = -\boldsymbol{\omega}^{\times}\mathbf{r} + \mathbf{v} + \mathbf{n}_r, \tag{4.41}$$

$$\dot{\mathbf{v}} = -\boldsymbol{\omega}^{\times}\mathbf{v} + \mathbf{a} + \mathbf{q}^{-1}(\mathbf{g}), \tag{4.42}$$

$$\dot{\mathbf{q}} = -\mathbf{q}(\boldsymbol{\omega}), \tag{4.43}$$

$$\dot{\mathbf{b}}_a = \mathbf{n}_{b_a}, \tag{4.44}$$

$$\dot{\mathbf{b}}_g = \mathbf{n}_{b_g}, \tag{4.45}$$

$$\dot{\mathbf{c}} = \mathbf{n}_c, \tag{4.46}$$

$$\dot{\mathbf{z}} = \mathbf{n}_z, \tag{4.47}$$

$$\dot{\mathbf{n}(\boldsymbol{\mu}_i)} = \mathbf{N}(\boldsymbol{\mu}_i)^T \left(\boldsymbol{\omega}_{\mathcal{C}} + \mathbf{n}(\boldsymbol{\mu}_i)^{\times} \frac{\mathbf{v}_{\mathcal{C}}}{d(\rho_i)} \right) + \mathbf{n}_{\mu,i}, \tag{4.48}$$

$$\dot{\rho}_i = \mathbf{n}(\boldsymbol{\mu}_i)^T \frac{\mathbf{v}_{\mathcal{C}}}{d'(\rho_i)} + n_{\rho,i}, \tag{4.49}$$

where $\mathbf{N}(\boldsymbol{\mu}_i)^T$ is used to project a 3D vector onto the tangent space of the vector $\mathbf{n}(\boldsymbol{\mu}_i)$. The gravity vector $\mathbf{g}$ has to be rotated, since it is expressed in the world frame. All the terms with symbol n_* are white Gaussian noises, vectorial and scalar, with different dimensions.
Analyzing the dimensions of the derivative vector of bearing vector and rotation it can be seen that a consistent minimal representation of the state is accomplished. In fact they respectively belong to a 2D and 3D vector space.
The continuous time equation can be discretized by a forward Euler method, considering the use of the boxplus operator when needed. The resulting discrete equation can then be used in the filter framework for the prediction step. The non trivial results are reported next:

$$\begin{aligned}\mathbf{q}_{k+1} &= \mathbf{q}_k \boxplus (-\Delta t\ \mathbf{q}_k(\boldsymbol{\omega}_k)) = exp((-\Delta t\ \mathbf{q}_k(\boldsymbol{\omega}_k))) \otimes \mathbf{q}_k \\ &= \mathbf{q}_k \otimes exp((-\Delta t\ \boldsymbol{\omega}_k)) \otimes \mathbf{q}_k^{-1} \otimes \mathbf{q}_k \\ &= \mathbf{q}_k \otimes exp(\Delta t\ (\mathbf{b}_{g,k} + \mathbf{n}_{\omega,k} - \tilde{\boldsymbol{\omega}}_k)),\end{aligned} \tag{4.50}$$

$$\mathbf{z}_{k+1} = \mathbf{z}_k \boxplus \Delta t\ \mathbf{n}_{z,k} = exp(\Delta t\ \mathbf{n}_{z,k}) \otimes \mathbf{z}_k, \tag{4.51}$$

$$\boldsymbol{\mu}_{i,k+1} = exp\left(\Delta t\left((\mathbf{I} - \mathbf{n}(\boldsymbol{\mu}_{i,k})\mathbf{n}(\boldsymbol{\mu}_{i,k})^T)\boldsymbol{\omega}_{\mathcal{C}} + \mathbf{n}(\boldsymbol{\mu}_{i,k})^{\times}\frac{\mathbf{v}_{\mathcal{C}}}{d(\rho_{i,k})} + \mathbf{N}(\boldsymbol{\mu}_{i,k})\mathbf{n}_{\mu,i}\right)\right) \otimes \boldsymbol{\mu}_{i,k}. \tag{4.52}$$

Usually in system with both camera and IMU, the latter sends measurement at a much higher rate, and this would lead to recompute the propagation step frequently. The resulting computational load can be unbearable, especially for embedded systems. To circumvent the issue it is applied a IMU pre-integration technique, that consist in using an averaged value of the measurements correct just with a linearly varying bias (sec. 4.3.2). In ROVIO it is implemented a simpler solution where the Jacobian (4.31) is evaluated using the averaged IMU values every camera measurement only.

4.2.10 Direct innovation term and update

Inspecting eq. 4.26 it is clear that the photomeric error, for a given image I and patch P_i depends exclusively on the estimation of the feature pixel coordinates $\mathbf{p}_i = \boldsymbol{\pi}_c(\mathbf{n}(\boldsymbol{\mu}_i))$. This means that the multilevel error, composed 72 terms, is only a function of the 2 dimensional bearing vector. It is therefore applied a QR decomposition to reduce the error to a more computationally manageable 2 dimensions. The QR decomposition or factorization technique transforms a square matrix in a equivalent product of a orthogonal and upper triangular matrix: $\mathbf{A} = \mathbf{QR}$. Different numerical implementations exist with a trade-off between numerical stability and execution time [40]. When it is applied to the gradient matrix it results in:

$$\begin{aligned}\mathbf{A}(\mathbf{p}_i, I, \mathbf{D}_i) &= \mathbf{Q}(\mathbf{p}_i, I, \mathbf{D}_i)\mathbf{R}(\mathbf{p}_i, I\mathbf{D}_i) \\ &= \begin{bmatrix} \mathbf{Q}_1(\mathbf{p}_i, I, \mathbf{D}_i) & \mathbf{Q}_2(\mathbf{p}_i, I, \mathbf{D}_i) \end{bmatrix} \begin{bmatrix} \mathbf{R}_1(\mathbf{p}_i, I, \mathbf{D}_i) \\ \mathbf{0} \end{bmatrix},\end{aligned} \tag{4.53}$$

where the matrix $\mathbf{R}_1(\mathbf{p}_i, I, \mathbf{D}_i)$ is upper triangular matrix with full rank 2 in case of corner features, rank 1 for line features and rank close to 0 for uniform patches. From the factorization it is derived the innovation term for a patch i and at the iteration step j:

$$\mathbf{y}_{i,j} = \mathbf{Q}_1(\boldsymbol{\pi}_c(\mathbf{n}(\boldsymbol{\mu}_{i,j}^+)), I, \mathbf{D}_i)^T \mathbf{b}(\boldsymbol{\pi}_c(\mathbf{n}(\boldsymbol{\mu}_{i,j}^+)), I, \mathbf{D}_i). \tag{4.54}$$

Its dimensions are related to the rank of the above triangular matrix. This design of the innovation term implicitly assumes the same noise for all the photomertic error terms. To generalize the formulation it could be used a weighting matrix. The reduced innovation Jacobian is:

$$\mathbf{H}_{i,j} = \mathbf{R}_1(\boldsymbol{\pi}_c(\mathbf{n}(\boldsymbol{\mu}_{i,j}^+)), I, \mathbf{D}_i) \frac{d\boldsymbol{\pi}_c}{d\mathbf{n}(\boldsymbol{\mu})}(\mathbf{n}(\boldsymbol{\mu}_{i,j}^+)) \tag{4.55}$$

4.2.11 Landmark tracking

The landmark tracking process is already included in the filter, by its iterative design. In fact, the update iterations at the same time align the patches with the new image frame and distribute the information in the states.
When a landmark has large associated uncertainty, a set of multiple pixel location hypothesis is created, all inside the uncertainty boundaries. Then it is performed an update iteration step to integrate the data from IMU and process model with the photometric error. Outlier rejection techniques are finally applied to select the best landmark to track.
This implementation has also the advantage of properly tracking line features (the innovation only affects the direction perpendicular to the line) and smooth patches (the state vector does not change and the iterations stop). A visual representation is reported in figure 4.4.

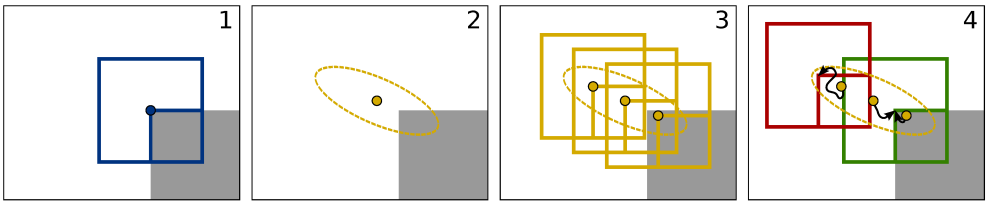


FIGURE 4.4: ROVIO landmark tracking. The landmark from the previous iteration (blue square) is used to estimate the new position in the image frame, with a corresponding covariance (yellow ellipse). Multiple candidates of the new landmark location are created (yellow dots in 3) and from a iteration step and outlier rejection the best candidate is extracted (green square). [35].

4.2.12 Landmark management

The filter implementation limits the number of landmarks that can be tracked simultaneously, putting a lot of importance on their quality. In addition to the scoring explained in section 4.2.6 used to choose the feature, an the outlier rejection is applied. It consists of multiple techniques:

- Mahalanobis distance on the innovation residual: it is used to remove moving objects and image disturbances, by computing the Mahalanobis distance of between the innovation residual, $\mathbf{r}_k = \mathbf{H}_{k,n}\mathbf{L}_{k,n}(\mathbf{x}_{k,n}^+ \boxminus \mathbf{x}_k^-) - \mathbf{h}(\mathbf{x}_{k,n}^+, \mathbf{0})$ (4.35), and the predicted innovation covariance, $\mathbf{P}_k^+$ (4.36), that is written as: $\|\mathbf{r}_k\|^2_{\mathbf{P}_k^+} = \mathbf{r}_k^T \mathbf{P}_k^{+^{-1}} \mathbf{r}_k$. When the result is bigger then a threshold, the measure is excluded. Using this method the image gradient is intrinsically accounted for, causing the faster rejection of smooth image patches.
  The Mahalanobis distance is a generalization of the Euclidean distance in case of normal distributed variables. It allows to find the distance between a multivariate random variable and a multivariate normal distribution, by weighting the distance from the distribution mean by its covariance [41],

- Threshold on the total photometric error: when the error exceeds a set value, the patch is considered to not be matched correctly with the image subsection and therefore excluded,

- Feature quality check: it is implemented by assessing the innovation residual of 4 close locations. If at least two of this points have a fairly larger residual then the feature, the tracking is good.

To take in account the landmark proprieties in time, a quality score is maintained. This is based on the combination of three scores:

- Global quality: expresses the time span in which the landmark has been tracked from its initialization,

- Local quality: outlines how frequently the landmark has been tracked in the field of view of recent frames, when it was expected to be there,

- Local visibility: shows the amount of time the landmark was visible in recent frames.

The score is used with an adaptive thresholding to guide the landmarks removal and control their numbers.

4.2.13 Stereo camera setup

The framework implementation with a stereo camera has little differences compared to what explained so far. ROVIO still expresses the landmarks in a single camera frame, the detection frame. If multiple measurements of the same landmark come at the same

time from different point of view, stereo triangulation is applied, to find the initial distance. Once the initialization is over, the landmark is projected in both camera frames at every time step. If landmark is observed in different camera frame from the one in which it is parameterized, its bearing vector has to be transform before computing the iterative innovation. The transformation is:

$$\mathbf{n}(\boldsymbol{\mu}_m, i) = \mathbf{z}_m(\mathbf{c}_d + \mathbf{z}_d^{-1}(\mathbf{n}(\boldsymbol{\mu}_d, i)\ d(\rho_i)) - \mathbf{c}_m) \tag{4.56}$$

where $\mathbf{z}_*$ and $\mathbf{c}_*$ represent the rotational and linear camera-IMU extrinsics. While m stands for the measurement camera frame and d for the detection camera frame. Moreover, the Jacobian in eq. 4.55 must be right-multiplied by the Jacobian of the transformation.

4.3 VINS-Fusion

The material that is not explicitly referenced in this section 4.2 about VINS-Fusion comes from [42], [43], [44] and from direct analysis of the source code, due to the lack of in-depth explanation of the VIO-only operation. Visual INertial System - Fusion is a general non-linear optimization based framework that can fuse together local estimations (VIO) and global sensor measurements (e.g GPS). Optimization-based methods can maintain a lot of measurements and optimize multiple variables at once, also known as Bundle Adjustment (BA). They have advantage in time synchronization, compared with filter-based methods. The big bundle they use serves as a nature buffer, therefore they can easily handle the case when measurements from multiple sensors come in disorder. Optimization-based algorithms also outperform the filter-based algorithms in term of accuracy, but at the cost of computational complexity. The VIO at the base of this system is an evolution of VINS-Mono, for more general cases:

- Monocular camera and IMU
- Stereo camera and IMU
- Stereo camera only

The system is composed by three main modules:

- VO-VIO module: it consists in a sliding window and tightly-coupled VIO, used to obtain higly accurate and robust state estimation. It performs online estimation of the IMU bias and it can also refine the camera-IMU extrinsics and estimate the camera-IMU timing difference. The world frame is set in the first IMU measurement.
- Local pose estimation: using only local sensors (sec. 4.3.1) and the pose estimate coming from the first module, it creates a pose graph with keyframes. BRIEF descriptors extracted in these keyframes are used, in conjunction with DBoW2

state-of-the-art bag-of-word place recognition approach, to find a loop closure. Once the loop is detected, the entire pose graph is optimized, resulting in a drift-free pose estimation.

- Global pose estimation: is the same procedure of the local pose estimation, but with local and global sensors (sec 4.3.1). The world frame is the same fixed frame used by global sensors.

Since the interesting case for this thesis is the use of stereo VIO, only this one is explained in detail.

4.3.1 Local and global sensors

The sensors can be categorized accordingly to the measurement reference frame:

- Local sensors: Camera, LiDAR, IMU, etc. These sensors don't have a global reference, thus usually the first pose of the robot is set as the origin. The estimation of the robot's pose evolves incrementally from there, accumulating drift that will grow increasing the traveled distance.
- Global sensors: GPS, magnetometer, barometer, etc. These sensors work in a fixed global reference frame (e.g. the earth frame). The origin of this frame is known in advance and fixed in time. The measurement error is independent on the traveled distance.

4.3.2 Stereo VIO module

It is implemented by a sliding window and tightly-coupled VIO as shown in figure 4.5.

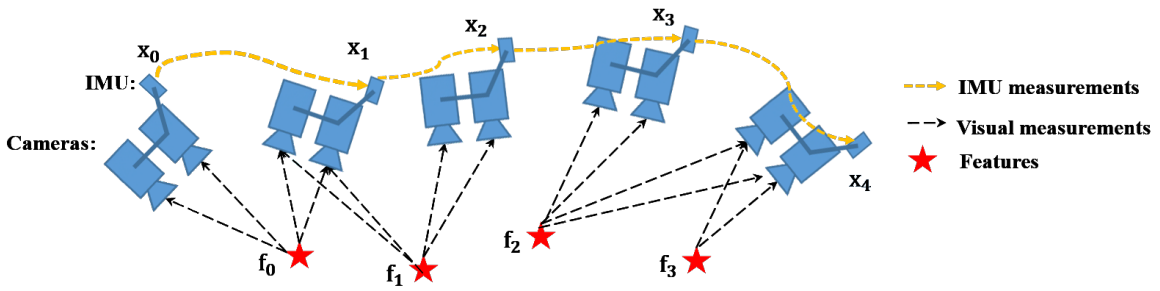


FIGURE 4.5: VINS-Fusion framework visual representation. Image adapted from [43].

Vision pre-processing

Existing 2D features are tracked in every new image coming from the left camera, using a KLT sparse optical flow algorithm (sec. 3.4). In the meantime, new corner features are detected on the same image, using Shi-Tomasi Corner Detector (sec. 3.3.2) and a

correspondent ones are found on the right image, applying again the KLT algorithm. To select only the more robust pairs, the KLT optical flow is applied to the right camera features, to get the correspondent left ones, and the features too far from the originals are discarded. This results in pairs of features that can be used to find their depth by triangulation. The corner detector assures a uniform feature distribution by adopting a minimum pixel distance between two neighboring features. The features are first undistorted and then projected on the image plane. This is achieved using the chosen camera model π_c with offline calibrate parameters.

IMU pre-integration

In optimized-based algorithms, every time the poses are adjusted, it is required to re-propagate IMU measurements between them. This is quite demanding of computing power, so to avoid that it is used a pre-integration algorithm. The raw gyroscope and accelerometer measurements at a time t are modeled as:

$$\begin{aligned} \tilde{\mathbf{a}}_t &= \mathbf{a}_t + \mathbf{b}_{a_t} + \mathbf{R}_{\mathcal{W}}^{\mathcal{B}_t}\mathbf{g} + \mathbf{n}_a \\ \tilde{\boldsymbol{\omega}}_t &= \boldsymbol{\omega}_t + \mathbf{b}_{g_t} + \mathbf{n}_\omega \end{aligned} \tag{4.57}$$

The IMU measurement, which are expressed in the body frame, are a combination of:

- Platform dynamics
- Acceleration bias $\mathbf{b}_a$ and gyroscope bias $\mathbf{b}_g$
- Gravity acceleration, which is expressed in the world reference frame
- Additive noise, which is assumed to be Gaussian: $\mathbf{n}_a \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_a^2)$ and $\mathbf{n}_\omega \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_\omega^2)$

Acceleration bias and gyroscope bias are modeled as random walk, that has Gaussian derivatives, $\mathbf{n}_{b_a} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_{b_a}^2)$, $\mathbf{n}_{b_g} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_{b_g}^2)$:

$$\begin{aligned} \dot{\mathbf{b}}_{a_t} &= \mathbf{n}_{b_a} \\ \dot{\mathbf{b}}_{g_t} &= \mathbf{n}_{b_g} \end{aligned} \tag{4.58}$$

Considering two time instants, corresponding to image frames $\mathbf{b}_k$ and $\mathbf{b}_{k+1}$ position, velocity, and orientation states, in the world frame, can be propagated by inertial measurements during the time interval $[t_k,\ t_{k+1}]$:

$$\begin{aligned} \mathbf{p}_{\mathcal{B}_{k+1}}^{\mathcal{W}} &= \mathbf{p}_{\mathcal{B}_k}^{\mathcal{W}} + \mathbf{v}_{\mathcal{B}_k}^{\mathcal{W}}\Delta t_k + \iint_{t\in[t_k,\ t_{k+1}]} (\mathbf{R}_{\mathcal{B}_t}^{\mathcal{W}}(\tilde{\mathbf{a}}_t - \mathbf{b}_{a_t} - \mathbf{n}_a) - {}_{\mathcal{W}}\mathbf{g})dt^2 \\ \mathbf{v}_{\mathcal{B}_{k+1}}^{\mathcal{W}} &= \mathbf{v}_{\mathcal{B}_k}^{\mathcal{W}} + \int_{t\in[t_k,\ t_{k+1}]} (\mathbf{R}_{\mathcal{B}_t}^{\mathcal{W}}(\tilde{\mathbf{a}}_t - \mathbf{b}_{a_t} - \mathbf{n}_a) - {}_{\mathcal{W}}\mathbf{g})dt \\ \mathbf{q}_{\mathcal{B}_{k+1}}^{\mathcal{W}} &= \mathbf{q}_{\mathcal{B}_k}^{\mathcal{W}} \otimes \int_{t\in[t_k,\ t_{k+1}]} \frac{1}{2}\boldsymbol{\Omega}(\tilde{\boldsymbol{\omega}}_t - \mathbf{b}_{g_t} - \mathbf{n}_{\boldsymbol{\omega}})\mathbf{q}_{\mathcal{B}_t}^{\mathcal{B}_k} dt \end{aligned} \tag{4.59}$$

and, considering the vectors as column vectors, with

$$\mathbf{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} -\boldsymbol{\omega}^{\times} & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^{T} & 0 \end{bmatrix} \tag{4.60}$$

As demonstrated by the equations, the IMU state propagation requires rotation, position and velocity of frame b_k. Therefore every time this states change, IMU measurements re-propagation is required. To avoid it, the pre-integration algorithm is applied. It consists in pre-integrating the parts which are related to linear acceleration $\tilde{\mathbf{a}}$ and angular velocity $\tilde{\boldsymbol{\omega}}$, after having transformed the reference frame from the world to the local body frame $\mathcal{B}_k$. This leads to:

$$\begin{aligned} \mathbf{R}_{\mathcal{W}}^{\mathcal{B}_k}\mathbf{p}_{\mathcal{B}_{k+1}}^{\mathcal{W}} &= \mathbf{R}_{\mathcal{W}}^{\mathcal{B}_k}(\mathbf{p}_{\mathcal{B}_k}^{\mathcal{W}} + \mathbf{p}_{\mathcal{B}_k}^{\mathcal{W}}\Delta t_k - \frac{1}{2}{}_{\mathcal{W}}\mathbf{g}\Delta t_k^2) + \boldsymbol{\alpha}_{\mathcal{B}_{k+1}}^{\mathcal{B}_k} \\ \mathbf{R}_{\mathcal{W}}^{\mathcal{B}_k}\mathbf{v}_{\mathcal{B}_{k+1}}^{\mathcal{W}} &= \mathbf{R}_{\mathcal{W}}^{\mathcal{B}_k}(\mathbf{v}_{\mathcal{B}_k}^{\mathcal{W}} + {}_{\mathcal{W}}\mathbf{g}\Delta t_k) + \boldsymbol{\beta}_{\mathcal{B}_{k+1}}^{\mathcal{B}_k} \\ \mathbf{q}_{\mathcal{W}}^{\mathcal{B}_k} \otimes \mathbf{q}_{\mathcal{W}}^{\mathcal{B}_{k+1}} &= \boldsymbol{\gamma}_{\mathcal{B}_{k+1}}^{\mathcal{B}_k} \end{aligned} \tag{4.61}$$

where,

$$\begin{aligned} \boldsymbol{\alpha}_{\mathcal{B}_{k+1}}^{\mathcal{B}_k} &= \iint_{t\in[t_k,\ t_{k+1}]} \mathbf{R}_{\mathcal{B}_t}^{\mathcal{B}_k}(\tilde{\mathbf{a}}_t - \mathbf{b}_{a_t} - \mathbf{n}_a)dt^2 \\ \boldsymbol{\beta}_{\mathcal{B}_{k+1}}^{\mathcal{B}_k} &= \int_{t\in[t_k,\ t_{k+1}]} \mathbf{R}_{\mathcal{B}_t}^{\mathcal{B}_k}(\tilde{\mathbf{a}}_t - \mathbf{b}_{a_t} - \mathbf{n}_a)dt \\ \boldsymbol{\gamma}_{\mathcal{B}_{k+1}}^{\mathcal{B}_k} &= \int_{t\in[t_k,\ t_{k+1}]} \frac{1}{2}\mathbf{\Omega}(\tilde{\boldsymbol{\omega}}_t - \mathbf{b}_{\boldsymbol{\omega}_t} - \mathbf{n}_g)\boldsymbol{\gamma}_{\mathcal{B}_t}^{\mathcal{B}_k}dt \end{aligned} \tag{4.62}$$

As shown, the pre-integration terms can be obtained exclusively using IMU measurements, considering $\mathcal{B}_k$ as reference frame, so they have nothing to do with the absolute position in the world reference frame. These terms are only related to IMU biases and not others states at time k and $k+1$. If the estimation of bias changes by a small amount, the coefficients $\boldsymbol{\alpha}_{\mathcal{B}_{k+1}}^{\mathcal{B}_k}, \boldsymbol{\beta}_{\mathcal{B}_{k+1}}^{\mathcal{B}_k}, \boldsymbol{\gamma}_{\mathcal{B}_{k+1}}^{\mathcal{B}_k}$ are adjusted by their first-order approximation with respect to the bias, while otherwise it is performed the re-propagation.
To implement this in discrete time, different numerical integration methods can be used. In the code implementation it is applied the mid-point integration, but to show the procedure is chosen the Euler method. At the beginning, $\boldsymbol{\alpha}_{\mathcal{B}_k}^{\mathcal{B}_k} = \boldsymbol{\beta}_{\mathcal{B}_k}^{\mathcal{B}_k} = 0$ and $\boldsymbol{\gamma}_{\mathcal{B}_k}^{\mathcal{B}_k}$ is the identity quaternion. The additive noise terms $\mathbf{n}_a$ and $\mathbf{n}_g$ is unknown and treated as zero. The mean of the pre-integration terms, in eq. 4.62, is propagated as follows:

$$\begin{aligned} \hat{\boldsymbol{\alpha}}_{\mathcal{B}_{i+1}}^{\mathcal{B}_k} &= \hat{\boldsymbol{\alpha}}_{\mathcal{B}_i}^{\mathcal{B}_k} + \hat{\boldsymbol{\beta}}_{\mathcal{B}_i}^{\mathcal{B}_k}\Delta t + \frac{1}{2}\mathbf{R}(\hat{\boldsymbol{\gamma}}_{\mathcal{B}_i}^{\mathcal{B}_k})(\tilde{\mathbf{a}}_i - \mathbf{b}_{a_i})\Delta t^2 \\ \hat{\boldsymbol{\beta}}_{\mathcal{B}_{i+1}}^{\mathcal{B}_k} &= \hat{\boldsymbol{\beta}}_{\mathcal{B}_i}^{\mathcal{B}_k} + \mathbf{R}(\hat{\boldsymbol{\gamma}}_{\mathcal{B}_i}^{\mathcal{B}_k})(\tilde{\mathbf{a}}_i - \mathbf{b}_{a_i})\Delta t \\ \hat{\boldsymbol{\gamma}}_{\mathcal{B}_{i+1}}^{\mathcal{B}_k} &= \hat{\boldsymbol{\gamma}}_{\mathcal{B}_i}^{\mathcal{B}_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}(\tilde{\boldsymbol{\omega}}_i - \mathbf{b}_{g_i}) \end{bmatrix} \end{aligned} \tag{4.63}$$

the values marked by a hat are the estimated ones and i is a discrete instant corresponding to a IMU measurement with $i \in [t_k,\ t_{k+1}]$. Δt is the time interval between two IMU measurements i and $i+1$.
Considering $t \in [k, k+1]$, the four-dimensional rotation quaternion $\boldsymbol{\gamma}_{\mathcal{B}_{k+1}}^{\mathcal{B}_k}$ is over-parameterized, thus its error term is defined as a variation around its mean:

$$\boldsymbol{\gamma}_{\mathcal{B}_t}^{\mathcal{B}_k} \approx \hat{\boldsymbol{\gamma}}_{\mathcal{B}_t}^{\mathcal{B}_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\Delta\boldsymbol{\theta}_{\mathcal{B}_t}^{\mathcal{B}_k} \end{bmatrix} \tag{4.64}$$

where $\delta\boldsymbol{\theta}_{\mathcal{B}_t}^{\mathcal{B}_k}$ is a three-dimensional small perturbation. The continuous-time linearized dynamics of error terms of eq.4.63 is:

$$\begin{aligned} \begin{bmatrix} \delta\dot{\boldsymbol{\alpha}}_{\mathcal{B}_t}^{\mathcal{B}_k} \\ \delta\dot{\boldsymbol{\beta}}_{\mathcal{B}_t}^{\mathcal{B}_k} \\ \delta\dot{\boldsymbol{\theta}}_{\mathcal{B}_t}^{\mathcal{B}_k} \\ \delta\dot{\mathbf{b}}_{a_t} \\ \delta\dot{\mathbf{b}}_{g_t} \end{bmatrix} &= \begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\mathbf{R}_{\mathcal{B}_t}^{\mathcal{B}_k}(\tilde{\mathbf{a}}_t - \mathbf{b}_{a_t})^{\times} & -\mathbf{R}_{\mathcal{B}_t}^{\mathcal{B}_k} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -(\tilde{\boldsymbol{\omega}}_t - \mathbf{b}_{g_t})^{\times} & \mathbf{0} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{\alpha}_{\mathcal{B}_t}^{\mathcal{B}_k} \\ \delta\boldsymbol{\beta}_{\mathcal{B}_t}^{\mathcal{B}_k} \\ \delta\boldsymbol{\theta}_{\mathcal{B}_t}^{\mathcal{B}_k} \\ \delta\mathbf{b}_{a_t} \\ \delta\mathbf{b}_{g_t} \end{bmatrix} \\ &+ \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\mathbf{R}_{\mathcal{B}_t}^{\mathcal{B}_k} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{n}_a \\ \mathbf{n}_g \\ \mathbf{n}_{b_a} \\ \mathbf{n}_{b_g} \end{bmatrix} = \mathbf{F}_{\mathcal{B}_t}\delta\mathbf{z}_{\mathcal{B}_t}^{\mathcal{B}_k} + \mathbf{G}_{\mathcal{B}_t}\mathbf{n}_{\mathcal{B}_t} \end{aligned} \tag{4.65}$$

The covariance matrix $\mathbf{P}_{\mathcal{B}_{k+1}}^{\mathcal{B}_k}$ can be recursively computed using the first-order discrete-time covariance update with initial value $\mathbf{P}_{\mathcal{B}_k}^{\mathcal{B}_k} = \mathbf{0}$:

$$\mathbf{P}_{\mathcal{B}_{t+\Delta t}}^{\mathcal{B}_k} = (\mathbf{I} + \mathbf{F}_{\mathcal{B}_t}\Delta t)\mathbf{P}_{\mathcal{B}_t}^{\mathcal{B}_k}(\mathbf{I} + \mathbf{F}_{\mathcal{B}_t}\Delta t)^T + (\mathbf{G}_{\mathcal{B}_t}\Delta t)\mathbf{Q}(\mathbf{G}_{\mathcal{B}_t}\Delta t)^T \tag{4.66}$$

where $\mathbf{Q}$ is the noise covariance matrix:

$$\mathbf{Q} = \begin{bmatrix} \boldsymbol{\sigma}_a^2 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\sigma}_{\omega}^2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\sigma}_{b_a}^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{\sigma}_{b_g}^2 \end{bmatrix}, \tag{4.67}$$

where $\boldsymbol{\sigma}_*^2$ is the diagonal matrix with squared entries $\begin{bmatrix}\sigma_{*,x}^2 & \sigma_{*,y}^2 & \sigma_{*,z}^2\end{bmatrix}$ on the diagonal. The first-order Jacobian matrix $\mathbf{J}_{\mathcal{B}_{k+1}}$ of $\delta\mathbf{z}_{\mathcal{B}_{k+1}}^{\mathcal{B}_k}$ with respect to $\delta\mathbf{z}_{\mathcal{B}_k}^{\mathcal{B}_k}$ is also computed recursively using the initial value $\mathbf{J}_{\mathcal{B}_k} = \mathbf{I}$:

$$\mathbf{J}_{\mathcal{B}_{t+\delta t}} = (\mathbf{I} + \mathbf{F}_{\mathcal{B}_t}\delta t)\mathbf{J}_{\mathcal{B}_t} \tag{4.68}$$

The first order approximation of $\boldsymbol{\alpha}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}}$, $\boldsymbol{\beta}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}}$, $\boldsymbol{\gamma}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}}$ with respect to biases can be written as:

$$\begin{aligned}
\boldsymbol{\alpha}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}} &\approx \hat{\boldsymbol{\alpha}}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}} + \mathbf{J}^{\alpha}_{b_a}\delta\mathbf{b}_{a_k} + \mathbf{J}^{\alpha}_{b_g}\delta\mathbf{b}_{g_k} \\
\boldsymbol{\beta}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}} &\approx \hat{\boldsymbol{\beta}}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}} + \mathbf{J}^{\beta}_{b_a}\delta\mathbf{b}_{a_k} + \mathbf{J}^{\beta}_{b_g}\delta\mathbf{b}_{g_k} \\
\boldsymbol{\gamma}^{\mathcal{B}_k}_{\mathcal{B}_t} &\approx \hat{\boldsymbol{\gamma}}^{\mathcal{B}_k}_{\mathcal{B}_t} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\mathbf{J}^{\gamma}_{b_g}\delta\mathbf{b}_{g_k} \end{bmatrix}
\end{aligned} \tag{4.69}$$

where $\mathbf{J}^{\alpha}_{b_a}$ is the $\mathbf{J}^{\alpha}_{b_a}$ matrix a sub-block corresponding to the element $\dfrac{\delta\boldsymbol{\alpha}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}}}{\delta\mathbf{b}_{a_k}}$. The definition is analogous for $\mathbf{J}^{\alpha}_{b_g}$, $\mathbf{J}^{\beta}_{b_a}$, $\mathbf{J}^{\beta}_{b_g}$, $\mathbf{J}^{\gamma}_{b_g}$. When the estimation of bias varies a bit, eq.4.69 is used to correct pre-integration results instead of re-propagation. It is now possible to express the IMU measurement model with corresponding covariance matrix $\mathbf{P}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}}$:

$$\begin{bmatrix} \hat{\boldsymbol{\alpha}}^{\mathcal{B}_k}_{\mathcal{B}_t} \\ \hat{\boldsymbol{\beta}}^{\mathcal{B}_k}_{\mathcal{B}_t} \\ \hat{\boldsymbol{\theta}}^{\mathcal{B}_k}_{\mathcal{B}_t} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{R}^{\mathcal{B}_k}_{\mathcal{W}}(\mathbf{p}^{\mathcal{W}}_{\mathcal{B}_{k+1}} - \mathbf{p}^{\mathcal{W}}_{\mathcal{B}_k} + \frac{1}{2}\mathcal{W}\mathbf{g}\Delta t_k^2 - \mathbf{v}^{\mathcal{W}}_{\mathcal{B}_k}\Delta t_k) \\ \mathbf{R}^{\mathcal{B}_k}_{\mathcal{W}}(\mathbf{v}^{\mathcal{W}}_{\mathcal{B}_{k+1}} - \mathbf{v}^{\mathcal{W}}_{\mathcal{B}_k}) + \mathcal{W}\mathbf{g}\Delta t_k \\ \mathbf{q}^{\mathcal{W}\,-1}_{\mathcal{B}_k} \otimes \mathbf{q}^{\mathcal{W}}_{\mathcal{B}_{k+1}} \\ \mathbf{b}_{a_{k+1}} - \mathbf{b}_{a_k} \\ \mathbf{b}_{g_{k+1}} - \mathbf{b}_{g_k} \end{bmatrix} \tag{4.70}$$

Visual inertial estimator initialization

A bundle of images is maintained, with a sliding window, to limit the computation complexity. If enough features (4 in the latest implementation) are tracked in the current frame, they are used to initialize the camera pose by a triangulation and perspective-n-point (PnP) method. This method search a correspondence between a set of 2D points in the image frame and a set of 3D points in the world, using the camera calibration parameters to reconstruct its pose [45]. Once the camera pose is initialized, all the poses for the other frames in the windows are estimated. The gyroscope bias calibration is then performed. In the end, a global full bundle adjustments is applied to minimize the total reprojection error of all feature observations.

Gyroscope bias calibration

From the visual structure from motion, considering two consecutive frames $\mathbf{b}_k$ and $\mathbf{b}_{k+1}$, the correspondent rotations $\mathbf{q}^{\mathcal{C}_0}_{\mathcal{B}_k}$ and $\mathbf{q}^{\mathcal{C}_0}_{\mathcal{B}_{k+1}}$ are known. From the IMU pre-integration, the relative constraint $\hat{\boldsymbol{\gamma}}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}}$ is also known. The IMU pre-integration term is linearized with respect to the gyroscope bias, using the bias Jacobian obtained in section 4.68:

$$\boldsymbol{\gamma}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}} \approx \hat{\boldsymbol{\gamma}}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\mathbf{J}^{\gamma}_{\mathbf{b}_g}\delta\mathbf{b}_g \end{bmatrix} \tag{4.71}$$

Then it is found the minimum of this cost function:

$$\min_{\delta\mathbf{b}_g} \sum_{k\in F} \left\| \mathbf{q}^{\mathcal{C}_0}_{\mathcal{B}_{k+1}}{}^{-1} \otimes \mathbf{q}^{\mathcal{C}_0}_{\mathcal{B}_k} \otimes \boldsymbol{\gamma}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}} \right\|^2 \tag{4.72}$$

where F is the set of indexes of frames in the window. This minimization results in the initial calibration of the gyroscope bias $\mathbf{b}_g$. All IMU pre-integration terms $\hat{\boldsymbol{\alpha}}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}}, \hat{\boldsymbol{\beta}}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}}, \hat{\boldsymbol{\gamma}}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}}$ are then re-propagated, using the new gyroscope bias.

States

The full state vector in the sliding window is defined as:

$$\begin{aligned}
\mathcal{X} &:= \left[\mathbf{p}^{\mathcal{W}}_{\mathcal{B}_0},\ \mathbf{R}^{\mathcal{W}}_{\mathcal{B}_0}\ \mathbf{p}^{\mathcal{W}}_{\mathcal{B}_1}\ \mathbf{R}^{\mathcal{W}}_{\mathcal{B}_1}\ ...\ \mathbf{p}^{\mathcal{W}}_{\mathcal{B}_n}\ \mathbf{R}^{\mathcal{W}}_{\mathcal{B}_n}\ \mathbf{x}_{cam}\mathbf{x}_{imu}\right] \\
\mathbf{x}_{cam} &:= [\lambda_0\ \lambda_1\ ...\ \lambda_l] \\
\mathbf{x}_{imu} &:= \left[\mathbf{v}^{\mathcal{W}}_{\mathcal{B}_0},\ \mathbf{b}_{a_0}\ \mathbf{b}_{g_0}\ \mathbf{v}^{\mathcal{W}}_{\mathcal{B}_1}\ \mathbf{b}_{a_1}\ \mathbf{b}_{g_1}\ ...\ \mathbf{v}^{\mathcal{W}}_{\mathcal{B}_n}\ \mathbf{b}_{a_n}\ \mathbf{b}_{g_n}\right]
\end{aligned} \tag{4.73}$$

where $\mathbf{p}$ and $\mathbf{R}$ are the position and orientation of the body in the world frame. $\mathbf{x}_{cam}$ is a variable related to the camera and composed of the features in the sliding window, observed in the first frame. $\mathbf{x}_{imu}$ is a IMU-related state, that includes of velocity, accelerometer bias and gyroscope bias. The transformation between camera frame and body frame is supposed to be calibrated offline.

Cost function

As explained in [46], the state estimation consist in a Maximum Likelihood Estimation (MLE) problem. It consists in finding the maximum of a likelihood function defined as the joint probability distribution of robot poses over a period of time. Considering the assumption of independent measurements, the problem is written as:

$$\hat{\mathcal{X}} = \arg\max_{\mathcal{X}} \prod_{t=0}^{n} \prod_{k \in \mathbf{S}} p(\mathbf{z}^k_t | \mathcal{X}) \tag{4.74}$$

where $\mathbf{z}$ are the measurements, $\mathcal{X}$ the states and $\mathbf{S}$ is a set of measurements which can come from any sensor. In practice the log-likelihood function is used. So, assuming a Gaussian distributed measurements uncertainty $p(\mathbf{z}^k_t, \mathcal{X}) \sim \mathcal{N}(\bar{\mathbf{z}}^k_t, \mathbf{P}^k_t)$, the equation becomes:

$$\hat{\mathcal{X}} = \arg\max_{\mathcal{X}} \prod_{t=0}^{n} \prod_{k \in \mathbf{S}} \exp\left(-\frac{1}{2} \| \mathbf{z}^k_t - h^k_t(\mathcal{X}) \|^2_{\mathbf{P}^k_t}\right) = \arg\min_{\mathcal{X}} \sum_{t=0}^{n} \sum_{k \in \mathbf{S}} \| \mathbf{e}^k_t \|^2_{\mathbf{P}^k_t} \tag{4.75}$$

where $h(\cdot)$ is the sensor model and $\mathbf{e}^k_t = \mathbf{z}^k_t - h^k_t(\mathcal{X})$ is the sensor error or sensor factor

Camera factor

The camera factor is a reprojection process: a feature is reprojected from the first observation camera frame to the subsequent ones. It is defined as the residual:

$$\mathbf{e}_t^l = \mathbf{z}_t^l - h_t^l(\mathbf{p}_{\mathcal{B}_i}^{\mathcal{W}}, \mathbf{R}_{\mathcal{B}_i}^{\mathcal{W}}, \mathbf{p}_{\mathcal{B}_t}^{\mathcal{W}}, \mathbf{p}_{\mathcal{B}_t}^{\mathcal{W}}, \lambda_t) = \begin{bmatrix} u_t^l \\ v_t^l \end{bmatrix} - \pi_c(\mathbf{T}_{\mathcal{C}}^{\mathcal{B}^{-1}} \mathbf{T}_{\mathcal{B}_t}^{\mathcal{W}^{-1}} \mathbf{T}_{\mathcal{B}_i}^{\mathcal{W}} \mathbf{T}_{\mathcal{C}}^{\mathcal{B}} {\pi_c}^{-1}(\lambda_l, \begin{bmatrix} u_i^l \\ v_i^l \end{bmatrix})) \tag{4.76}$$

where $\begin{bmatrix} u_i^l & v_i^i \end{bmatrix}^T$ is the first observation of the feature l, on the image plane of the image i. $\begin{bmatrix} u_t^l & v_t^l \end{bmatrix}^T$ is the observation of the same feature l, on the image plane of the image t. π_c is the projection function, and ${\pi_c}^{-1}$ is the back-projection one. $\mathbf{T}$ is a homogeneous transformation matrix as defined in section 2.2.1. $\mathbf{T}_{\mathcal{C}}^{\mathcal{B}}$ is the extrinsic transformation from camera frame to body frame, which is calibrated offline.
This residual, also called reprojection error, has a covariance matrix $\mathbf{P}_t^l$, which is a constant value in pixel coordinate, depending on the camera's intrinsic calibration.
The camera factor is defined both for left and right camera.

IMU factor

Considering the IMU measurements between two sequential images at time k and $k+1$, from eq. 4.70 the residual for pre-integrated IMU measurement is defined as:

$$\mathbf{e}_{k+1}^{\text{IMU}} = \begin{bmatrix} \delta\boldsymbol{\alpha}_{\mathcal{B}_{k+1}}^{\mathcal{B}_k} \\ \delta\boldsymbol{\beta}_{\mathcal{B}_{k+1}}^{\mathcal{B}_k} \\ \delta\boldsymbol{\theta}_{\mathcal{B}_{k+1}}^{\mathcal{B}_k} \\ \delta\mathbf{b}_a \\ \delta\mathbf{b}_g \end{bmatrix} = \begin{bmatrix} \boldsymbol{\alpha}_{\mathcal{B}_{k+1}}^{\mathcal{B}_k} - \mathbf{R}_{\mathcal{W}}^{\mathcal{B}_k}(\mathbf{p}_{\mathcal{B}_{k+1}}^{\mathcal{W}} - \mathbf{p}_{\mathcal{B}_k}^{\mathcal{W}} + \frac{1}{2}{}_{\mathcal{W}}\mathbf{g}\Delta t_k^2 - \mathbf{v}_{\mathcal{B}_k}^{\mathcal{W}}\Delta t_k) \\ \boldsymbol{\beta}_{\mathcal{B}_{k+1}}^{\mathcal{B}_k} - \mathbf{R}_{\mathcal{W}}^{\mathcal{B}_k}(\mathbf{v}_{\mathcal{B}_{k+1}}^{\mathcal{W}} - \mathbf{v}_{\mathcal{B}_k}^{\mathcal{W}}) + {}_{\mathcal{W}}\mathbf{g}\Delta t_k \\ 2\left[\mathbf{q}_{\mathcal{B}_k}^{\mathcal{W}^{-1}} \otimes \mathbf{q}_{\mathcal{B}_{k+1}}^{\mathcal{W}} \otimes (\boldsymbol{\gamma}_{\mathcal{B}_{k+1}}^{\mathcal{B}_k})^{-1}\right]_{xyz} \\ \mathbf{b}_{a_{k+1}} - \mathbf{b}_{a_k} \\ \mathbf{b}_{g_{k+1}} - \mathbf{b}_{g_k} \end{bmatrix} \tag{4.77}$$

where $[\mathbf{q}]_{xyz}$, extracts the quaternion $\mathbf{q}$ vector part, since it is used a three dimensional error state representation of the quaternion $\boldsymbol{\gamma}_{\mathcal{B}_{k+1}}^{\mathcal{B}_k}$ (eq. 4.62). The other therms have the same meaning as in eq.4.70. The residuals includes accelerometer and gyroscope bias for online correction.

Other factors

The system is build to accept any other sensors, as long as the measurements are modeled as general residual factors and these factors are added into the cost function.

Optimization

Usually Gauss-Newton or Levenberg-Marquardt approaches are used to solve the non-linear least square problem presented in . This means that the cost function is linearized

with respect to an initial guess $\hat{\mathcal{X}}$ and becomes:

$$\arg\min_{\delta\mathcal{X}} \sum_{t=0}^{n} \sum_{k\in\hat{\mathbf{S}}} \|\mathbf{e}_t^k + \boldsymbol{J}_t^k \delta\mathcal{X}\|_{\mathbf{P}_t^k}^2 \tag{4.78}$$

where e is a sensor factor and J is its Jacobian matrix. It is used $\hat{\mathbf{S}}$ instead of $\mathbf{S}$ to highlight the possibility to expand the cost function with two factors, not strictly related to sensors. In fact, VINS-Fusion can perform online refinement of the body-camera extrinsics and camera-IMU online temporal calibration. Using the linearization approximation, this cost function has closed-form solution of $\delta\mathcal{X}$. For example, using Gauss-Newton algorithm, the solution is found as:

$$\underbrace{\sum\sum \mathbf{J}_t^{k^T} \mathbf{P}_t^{k^{-1}} \mathbf{J}_t^k}_{\mathbf{H}} \delta\mathcal{X} = \underbrace{-\sum\sum \mathbf{J}_t^{k^T} \mathbf{P}_t^{k^{-1}} \mathbf{e}_t^k}_{\mathbf{b}} \tag{4.79}$$

Then, the current state $\hat{\mathcal{X}}$ is updated with $\hat{\mathcal{X}} \oplus \delta\mathcal{X}$, with $\oplus$ defined as the plus operation on manifold for rotation. The procedure is iterated multiple times, until it converges. The actual implementation utilizes Ceres solver, to get stable and optimal results efficiently.

Marginalization

The computational complexity increases quadratically with the number of states, which grows in time. Marginalization is carried out to bound the computational complexity. The procedure converts previous measurements into a prior term, which retains past information. The set of states to be marginalized out is denoted as $\mathcal{X}_m$, and the set of remaining states is denoted as $\mathcal{X}_r$. Computing the sum of marginalized factors (eq.4.79) results in $\mathbf{H}$ and $\mathbf{b}$. Rearranging the states order leads to the following equation:

$$\begin{bmatrix} \mathbf{H}_{mm} & \mathbf{H}_{mr} \\ \mathbf{H}_{rm} & \mathbf{H}_{rr} \end{bmatrix} \begin{bmatrix} \delta\mathcal{X}_m \\ \delta\mathcal{X}_r \end{bmatrix} = \begin{bmatrix} \mathbf{b}_m \\ \mathbf{b}_r \end{bmatrix} \tag{4.80}$$

The Schur complement is used to perform the marginalization:

$$\underbrace{(\mathbf{H}_{rr} - \mathbf{H}_{rm}{\mathbf{H}_{mm}}^{-1}\mathbf{H}_{mr})}_{\mathbf{H}_p} \delta\mathcal{X}_r = \underbrace{\mathbf{b}_r - \mathbf{H}_{rm}{\mathbf{H}_{mm}}^{-1}\mathbf{b}_m}_{\mathbf{b}_p} \tag{4.81}$$

This results in a new prior $\mathbf{H}_p$, $\mathbf{b}_p$ for the remaining states. The information about marginalized states is converted into prior term without any loss. The sliding window amounts at 10 key-frames. When a new camera frame comes, if it is a key-frame (there are enough feature tracked from the last key-frame and their parallax is big enough) it will stay in the window and the oldest frame is marginalized out. Otherwise only the visual measurements are marginalized and the pre-integrated IMU measurements are kept. This data is used in the pre-integration process for the newest frame. This is shown in figure 4.6. The marginalization is performed to maintain sparsity of the

system. Once the prior information is computed, the posterior can be found applying Bayes' rule:

$$p(\mathcal{X}|\mathbf{z}) \propto p(\mathbf{z}|\mathcal{X})p(\mathcal{X}) \tag{4.82}$$

This makes the state estimation a maximum a-posteriori problem that is written as:

$$\begin{aligned}\hat{\mathcal{X}}_{m:n} &= \arg\max_{\mathcal{X}_{m:n}} \prod_{t=0}^{n} \prod_{k\in\mathbf{S}} p(\mathbf{z}_t^k|\mathcal{X}_{m:n})p(\mathcal{X}_{m:n}) \\ &= \arg\min_{\mathcal{X}} \sum_{t=0}^{n} \sum_{k\in\mathbf{S}} \|\mathbf{z}_t^k - h_t^k(\mathcal{X}_{m:n})\|_{\mathbf{P}_t^k}^2 + (\mathbf{H}_p\delta\mathcal{X}_{m:n} - \mathbf{b}_p)\end{aligned} \tag{4.83}$$

where $\mathcal{X}_{m:n}$ are the states in the sliding window (from time m to n). The previous equation (eq.4.83), only adds a prior term to the eq.4.75 and it is solved in the same way.

Failure detection module

This VIO system is robust to multiple challenging motions and environments, but failure is still unavoidable. Drastic illumination changes and aggressive motions can make the estimator fail, therefore a failure detection and recovery strategy is employed, based on these criteria:

- The estimated accelerometer bias norm is larger then 2.5 m/s^2
- The estimated gyroscope bias norm is larger then 1.0 rad/s

When a failure is detected, the system tries to initialize itself again, and once it is successful, run again from there.

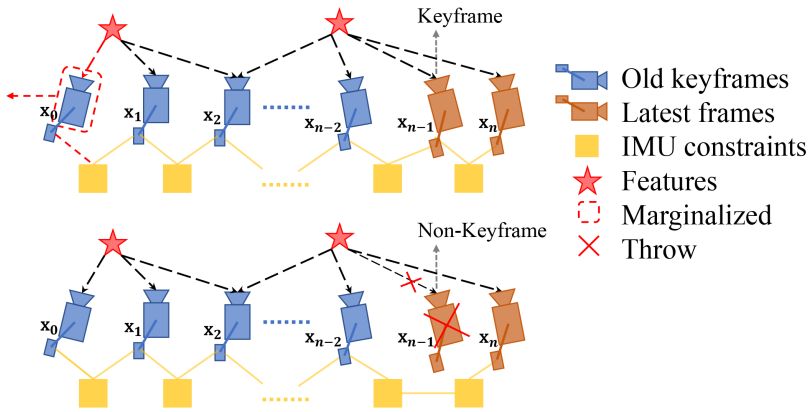


FIGURE 4.6: VINS-Fusion marginalization strategy. If the new camera frame is a keyframe it is kept in the window and the latest keyframe is marginalized out. Otherwise it is marginalized. The marginalization concerns only the cameras measurements, the pre-integrated inertial measurements are kept.[44].

4.4 OKVIS

The material that is not explicitly referenced in this section 4.3 about OKVIS comes from [47], and [48]. Open Keyframe-based Visual-Inertial SLAM is a thightly-coupled nonlinear and window-based optimization framework. The SLAM in the name does not stand for the classical sense of simultaneous localization and mapping in which a map is constructed and then optimized when the agent re-visits known places. In this case it describes the sliding window technique implemented by the filter, in which a limited number of key-points is maintained, so just a local map that keeps changing.

4.4.1 States

The state variables consist of the robot states at image times k, $\mathbf{x}_R^k$, and the landmarks at times k, $\mathbf{x}_L^k$. The first set of estimated variables is composed by the robot position in the inertial frame, $\mathbf{p}_{\mathcal{B}}^{\mathcal{W}}$, the body orientation quaternion $\mathbf{q}_{\mathcal{B}}^{\mathcal{W}}$, the velocity in the world frame, $\mathbf{v}_{\mathcal{B}}^{\mathcal{W}}$ and biases of the accelerometer $\mathbf{b}_a$ and of the gyroscope $\mathbf{b}_g$. The robot states at time k can therefore be written as:

$$\mathbf{x}_R^k := \left[\mathbf{p}_{\mathcal{B}_k}^{\mathcal{W}\,T}\ \mathbf{q}_{\mathcal{B}_k}^{\mathcal{W}\,T}\ \mathbf{v}_{\mathcal{B}_k}^{\mathcal{W}\,T}\ \mathbf{b}_{a_k}{}^T\ \mathbf{b}_{g_k}{}^T\right]^T \in \mathbb{R}^3 \times S(3) \times \mathbb{R}^9. \tag{4.84}$$

Two partition of the robot states are also used:

- the pose states, $\mathbf{x}_p := \left[\mathbf{p}_{\mathcal{B}}^{\mathcal{W}T}\ \mathbf{q}_{\mathcal{B}}^{\mathcal{W}T}\right]^T$
- the speed/bias states states, $\mathbf{x}_{sb} := \left[\mathbf{v}_{\mathcal{B}}^{\mathcal{W}T}\ \mathbf{b}_a{}^T\ \mathbf{b}_g{}^T\right]^T$

Using Euclidean coordinates when estimating a distant point with a Gauss-Newton estimator, will usually lead to the point being pushed towards infinity . Therefore, to allow a seamless integration of very close and very far landmarks, they are expressed in homogeneous coordinates with respect to the world frame: $\mathbf{x}_L := {}_{\mathcal{W}}\mathbf{l} := [l_x\ l_y\ l_z\ 1] \in \mathbb{R}^4$. Since in general the states reside in a manifold, it is used a perturbation in the tangent space $\mathfrak{g}$ of this manifold and employ the group operator $\boxplus$ and the exponential and logarithmic map. The perturbation is defined as: $\delta\mathbf{x} := \mathbf{x} \boxplus \hat{\mathbf{x}}^{-1}$ around the estimate $\hat{\mathbf{x}}$. A minimal coordinate representation is used, $\delta\mathcal{X} \in \mathbb{R}^{\dim\mathfrak{g}}$ and a bijective mapping $\Phi : \mathbb{R}^{\dim\mathfrak{g}} \to \mathfrak{g}$ that transforms from minimal coordinates to the tangent space. Thus, the transformations from and to minimal coordinates are:

$$\begin{aligned} \delta\mathbf{x} &= exp(\Phi(\delta\mathcal{X})) \\ \delta\mathcal{X} &= \Phi^{-1}(log(\delta\mathbf{x})) \end{aligned} \tag{4.85}$$

In practice, it is used the minimal 3D axis-angle perturbation of orientation $\delta\boldsymbol{\alpha} \in \mathbb{R}^3$, which can be converter in the equivalent quaternion $\delta\mathbf{q}$ using the exponential map:

$$\delta\mathbf{q} := exp\left(\begin{bmatrix} 0 \\ \frac{1}{2}\delta\boldsymbol{\alpha} \end{bmatrix}\right) = \begin{bmatrix} \cos\left(\|\frac{\delta\boldsymbol{\alpha}}{2}\|\right) \\ \text{sinc}\left(\|\frac{\delta\boldsymbol{\alpha}}{2}\|\right)\frac{\delta\boldsymbol{\alpha}}{2} \end{bmatrix}, \tag{4.86}$$

where sinc is the cardinal sine function, defined as

$$\text{sinc}(x) = \begin{cases} \dfrac{\sin(x)}{x} & \text{if } x \neq 0 \\ 1 & \text{if } x = 0 \end{cases}. \tag{4.87}$$

Using the group operator $\otimes$, $\mathbf{q}_{\mathcal{B}}^{\mathcal{W}} = \delta\mathbf{q} \otimes \hat{\mathbf{q}}_{\mathcal{B}}^{\mathcal{W}}$. The minimal robot error state vector is obtained as

$$\delta\mathcal{X}_R = \left[\delta\mathbf{p}^T \ \delta\boldsymbol{\alpha}^T \ \delta\mathbf{v}^T \ \delta\mathbf{b}_a{}^T \ \delta\mathbf{b}_g{}^T\right]^T \in \mathbb{R}^{15}, \tag{4.88}$$

and from it are analogously defined $\delta\mathcal{X}_p$ and $\delta\mathcal{X}_{sb}$. The homogeneous landmarks are considered as non-unit quaternions, with the minimal perturbation $\delta\boldsymbol{\beta}$, meaning that $\delta\mathcal{X}_L := \delta\boldsymbol{\beta}$.

4.4.2 Non-linear optimization

The problem formulation is the same as section 4.3.2, consisting in the minimization of a cost function (sections 4.3.2 and 4.3.2) containing visual and inertial error terms in the present window.

4.4.3 Reprojection error

A standard formulation of the reprojection error is used:

$$\mathbf{e}_l^{i,j,k} = \mathbf{z}^{i,j,k} - \mathbf{g}(\mathbf{T}_{\mathcal{W}\mathcal{W}}^{\mathcal{C}}\mathbf{l}^j) \tag{4.89}$$

where $(\cdot)^{i,j,k}$ is a vector expressed in the disparity space and $\mathbf{g}(\cdot)$ is the stereo projection model (sec 2.4.3)

4.4.4 IMU kinematics

Assuming the effect from Earth' rotation small compared to the gyroscope accuracy, the IMU model is defined similarly to section 4.3.2, as:

$$\begin{aligned}
\dot{\mathbf{p}}_{\mathcal{B}}^{\mathcal{W}} &= \mathbf{v}_{\mathcal{B}}^{\mathcal{W}}, \\
\dot{\mathbf{q}}_{\mathcal{B}}^{\mathcal{W}} &= \frac{1}{2}\boldsymbol{\Omega}\left({}_{\mathcal{B}}\tilde{\boldsymbol{\omega}}_{\mathcal{B}}^{\mathcal{W}} - \mathbf{n}_\omega - \mathbf{b}_g\right)\mathbf{q}_{\mathcal{B}}^{\mathcal{W}}, \\
\dot{\mathbf{v}}_{\mathcal{B}}^{\mathcal{W}} &= \mathbf{R}_{\mathcal{B}}^{\mathcal{W}}\left({}_{\mathcal{B}}\tilde{\mathbf{a}}_{\mathcal{B}}^{\mathcal{W}} - \mathbf{n}_a - \mathbf{b}_a\right) + {}_{\mathcal{W}}\mathbf{g}, \\
\dot{\mathbf{b}}_g &= \mathbf{n}_{b_g}, \\
\dot{\mathbf{b}}_a &= -\frac{1}{\tau}\mathbf{b}_a + \mathbf{n}_{b_a},
\end{aligned} \tag{4.90}$$

where $\mathbf{n}_* \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_{b_*}^2)$ and the other terms have a similar meaning to the aforementioned section. The main difference is in defining the gyroscope bias as a bounded

random walk with a time constant $\tau > 0$. The matrix $\mathbf{\Omega}$ comes from the estimated angular velocity $\boldsymbol{\omega}_{\mathcal{B}}^{\mathcal{W}}$ as shown in eq. 4.60. The linearized error dynamics consists in:

$$\delta\dot{\mathcal{X}}_R = \mathbf{F}_c(\mathcal{X}_R)\delta\mathcal{X}_R + \mathbf{G}(\mathbf{x}_R)\mathbf{n}, \tag{4.91}$$

where G is straightforward to derive and :

$$\mathbf{F}_c = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & [\mathbf{R}_{\mathcal{B}}^{\mathcal{W}}({}_{\mathcal{B}}\tilde{\mathbf{a}}_{\mathcal{B}}^{\mathcal{W}} - \mathbf{b}_a)]^{\times} & \mathbf{0} & \mathbf{0} & -\mathbf{R}_{\mathcal{B}}^{\mathcal{W}} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & -\frac{1}{\tau}\mathbf{I} \end{bmatrix} \tag{4.92}$$

To limit the computational complexity a simple Euler-Forward method is used for integration, resulting in the covariance propagation equation

$$\mathbf{P}_R^{p+1} = (\mathbf{I} + \mathbf{F}_c(\hat{\mathbf{x}}_R^p)\Delta t)\mathbf{P}_R^p(\mathbf{I} + \mathbf{F}_c(\hat{\mathbf{x}}_R^p)\Delta t)^T + (\mathbf{G}(\hat{\mathbf{x}}_R^p)\Delta t)\mathbf{Q}(\mathbf{G}(\hat{\mathbf{x}}_R^p)\Delta t)^T \tag{4.93}$$

where $\mathbf{Q}$ is the diagonal matrix containing all the noise covariances.

4.4.5 IMU measurement error term

Between two camera measurements, at time k and $k+1$, there are many IMU measurements that are in general not synchronized. Thus,the IMU error term must be a function of robot states at time k and $k+1$ and all the IMU measurements in between, so $\mathbf{e}_{IMU}^k(\mathbf{x}_R^k, \mathbf{x}_R^{k+1}, \mathbf{z}_{IMU}^{k+1})$. It is assumed an approximate normal conditional probability density f for given robot states:

$$f(\mathbf{e}_{IMU}^k|\mathbf{x}_R^k, \mathbf{x}_R^{k+1}) \approx \mathcal{N}(\mathbf{0}, \mathbf{P}_s^k). \tag{4.94}$$

Considering the state prediction $\hat{\mathbf{x}}_R^{k+1}(\mathbf{x}_R^k, \mathbf{z}_{IMU}^k)$ with covariance $\mathbf{P}\left(\delta\hat{\mathbf{x}}_R^{k+1}|\mathbf{x}_R^k, \mathbf{z}_{IMU}^k\right)$, the IMU prediction error is:

$$\mathbf{e}_{IMU}^k(\mathbf{x}_R^k, \mathbf{x}_R^{k+1}, \mathbf{z}_{IMU}^{k+1}) = \begin{bmatrix} \hat{\mathbf{p}}_{\mathcal{B}_{k+1}}^{\mathcal{W}} - \mathbf{p}_{\mathcal{B}_{k+1}}^{\mathcal{W}} \\ 2\left[\hat{\mathbf{q}}_{\mathcal{B}_{k+1}}^{\mathcal{W}} \otimes \mathbf{q}_{\mathcal{B}_{k+1}}^{\mathcal{W}\ -1}\right]_{x,y,z} \\ \hat{\mathbf{x}}_{sb}^{k+1} - \mathbf{x}_{sb}^{k+1} \end{bmatrix}, \tag{4.95}$$

simply the difference between the prediction based on the previous state and the actual state, apart for the orientation, that uses a multiplicative minimal error. Using the propagation law, the error covariance is:

$$\mathbf{P}_{IMU}^k = \frac{\partial \mathbf{e}_{IMU}^k}{\partial \delta\hat{\mathcal{X}}_R^{k+1}} \mathbf{P}\left(\delta\hat{\mathbf{x}}_R^{k+1}|\mathbf{x}_R^k, \mathbf{z}_{IMU}^k\right) \frac{\partial \mathbf{e}_{IMU}^k}{\partial \delta\hat{\mathcal{X}}_R^{k+1}} \tag{4.96}$$

4.4.6 Keypoint matching and keyframe selection

The image processing consists in a customized multiscale SSE-optimized Harris corner detector (sec. 3.3) combined with BRISK descriptor extraction (sec. 3.5). The detector enforces uniform keypoint distribution in the image by gradually suppressing corners with weaker score, as they are detected at a small distance from a stronger corner. The descriptors are oriented during the extraction, in the gravity direction projected on the image. At first, keypoints are stereo-triangulated and inserted into a local map. When enough past frames, with included set of keypoints, are present, 3D-2D matching is preformed. From the current pose prediction obtained by IMU integration, all landmarks that should be visible are considered for brute-force descriptor matching. An outlier rejection is performed applying chi-square test in image coordinates using the pose prediction. In this way the application of a costly RANSAC is avoided. A bounded set of camera frames is maintained, consisting of poses with associated images taken at that time instant, and the landmarks visible in these images are kept in the local map. Two type of frame sets are considered:

- A collection of the most recent S frames in the temporal window,
- A set of N keyframes spanning to the first moment

The keyframe selection is enforced by a simple heuristic: the frame is labeled as keyframe if the ratio between the image area spanned by matched points versus the area spanned by all detected points falls below 50-60%.

Partial marginalization

At the beginning the marginalized error term is determined using the first $N+1$ frames, $\mathbf{x}_T^{1,\dots,N+1}$ with corresponding speed and bias states $\mathbf{x}_{sb}^{1,\dots,N+1}$ as shown in figure . The N first frames are interpreted as keyframes and the corresponding speed and bias states are marginalized out. This is shown in figure 4.7.

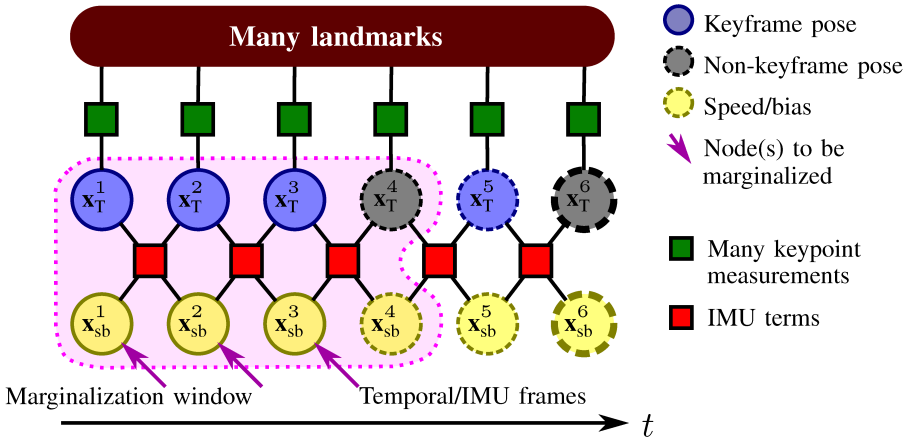


FIGURE 4.7: OKVIS: marginalization of the first frames. Speed and bias states outside the window are marginalized [48].

When a new frame $\mathbf{x}_T^c$ (c stands for current) is inserted in the optimization window, the marginalization operation is triggered. If the oldest frame in the temporal window, $\mathbf{x}_T^{c-S}$, is not a keyframe, its landmark measurements are erased and its speed and bias states are marginalized out. Eliminating landmarks is not the optimal solution but it keeps the problem sparse for a fast computation. This is shown in figure 4.8.

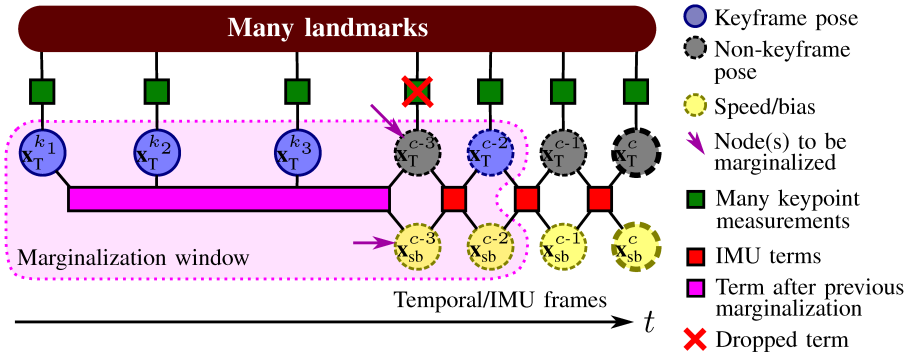


FIGURE 4.8: OKVIS: a regular frame is marginalized out [48].

If, instead, the oldest frame is a keyframe, it would be too significant to drop the landmarks, since all the relative pose information between the oldest two keyframes encoded in the common landmark observations would be lost. To avoid that all landmarks visible in the oldest keyframe but not in the most recent are marginalized out, as represented in figure 4.9. The mathematical formulation is the same as in section 4.3.2

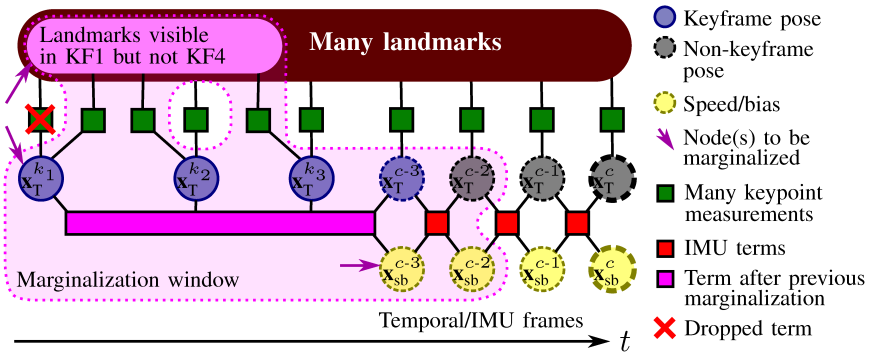


FIGURE 4.9: OKVIS: the oldest keyframe is marginalized out [48].

Chapter 5

Tests and results analysis

5.1 Tests

Jackal is a small all-terrain and weatherproof UGV. It features a IMU and a GPS, a top speed of 2 m/s and it lasts 4 hrs with basic usage. It has a powerful onboard computer and it comes fully equipped with ROS [49]. The one used also mounts a 2D LiDAR.

The camera was mounted on the Jackal using a custom 3D printed support as show in figure 5.1. The camera was tilted upwards by 25 degrees, since it seemed the best angle to get a good view of the environment. The camera software was run onboard the Jackal and the UGV was moved with a controller. The data was recorded in ROS bag format and then the VIO algorithms were run offline on a laptop. The data recorded was from the MYNTEYE camera and from the Jackal, consisting in the robot odometry resulted from a EKF fusing the robot wheel odometry and IMU measurements.

FIGURE 5.1: Jackal UGV with MYNTEYE camera.

In order to find the best setting of the algorithms parameters a few runs were recorded and the results were shown during the algorithms execution using RViz. RViz s a 3D visualizer for displaying data and state information form ROS. In this phase the Jackal odometry data was used as ground truth.
The camera fps were set to 20, because it seemed a good balance between information bandwidth and real-time operability.
Two cases were tested in two different indoor environments:

- Straight path in a corridor: the robot went straight ahead for about 15 meters, then rotated by 180 degrees and came back to the starting point, without rotating again,
- Pseudo-rectangular path in a room: the robot followed a path of about 16 meters, composed of straight movements and 90 degrees turns, to come back to the initial pose.

To make the experiment somewhat repeatable, the path was marked on the ground by tape and the UGV was guided with a gamepad to follow it. It was not easy to follow it along all the length by at a millimetric accuracy, but it was at least checked that the wheels passed on some control points with a lateral displacement of less then half of the wheel thickness (2.5 cm), otherwise the recording was discarded. The tests were repeated with two IMU frequency settings of 200 Hz and 500 Hz, and with the IR point projection turned on or off. Two run were preformed for each case, resulting in a total of 16 files.

The algorithms were set to not perform the online camera-IMU online time delay estimation, when available, because it was tested that it converged to an incorrect value and it made the localization worst. The online extrinsic camera-IMU estimation was disabled for the same reason. From multiple tests it was discovered for the logarithmic parametrization of the distance parameter in ROVIO (sec. 4.2.8) to be the best for in the chosen scenario, with the hyperbolic one closely in second place. The other methods lead to a considerable overestimation of the traveled distance. It was therefore used a logarithmic mapping. All algorithms initialize the world reference frame in the same manner: aligning z axis with the gravity vector, x axis opposite to the camera heading and y axis to complete the right-handed coordinate system.

5.2 Results analysis

The algorithms were run and the data saved in a ROS bag file, using modified roslaunch files and a bash script. First the ROS bag files were converted in text files for a faster loading, using an adaptation of the "bag to pose" script from the "rpg trajectory evaluation" repository [50]. Then, the files were then loaded in trajectory objects from the "trajectory toolkit" repository [51] and then modified with a custom python script. At the beginning the idea was to use the Jackal odometry as ground truth to evaluate the algorithms performance. It was difficult to start each trial from the exact same pose and the orientation misalignment is especially important, because impacts on the estimated position increasingly with the traveled distance. To account for it, the Jackal odometry trajectory were aligned with each other (in the same enviroment), considering the linear regression of the points in the first 20 cm of traveled distance. This realignment angle was also applied to the other odometries of the correspondent cases. Analyzing OKVIS and ROVIO trajectories it was found a common rotation about the vertical axis in all cases, possibly coming from a small tilt angle between the camera and Jackal reference frame. To account for it, all the VIO trajectories were rotated by 2 degrees counter-clockwise. VINS-Fusion instead shows a bigger common rotation angle of about 5 degrees, but clearly not resulting from physical rotation, instead maybe caused by an incorrect world frame initialization.
Once the data were plotted for each case, with also the ideal trajectory coming from measurements, it was evident that the Jackal odometry accuracy was in many cases comparable with the VIO ones, thus it was not possible to use it as ground truth.

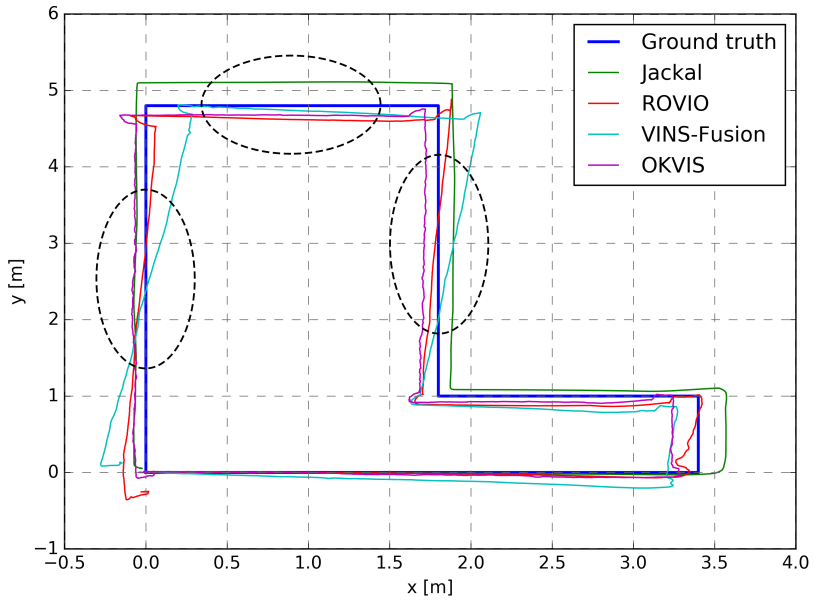


FIGURE 5.2: Trajectory comparison 200 Hz - IR off, first trial. The locations in which the UGV was following the ideal path at best are highlighted

It was chosen then to use what was certainly known, as a metrics:

- The relative error on the traveled length: $\delta_d = \dfrac{|\ d_m - d_{VIO}|}{d_m}$,
- The difference between initial and final position: $\Delta_p = \sqrt{(x_f - x_i)^2 + (y_f - y_i)^2}$,

where d_m is the traveled distance measured from the ideal path, d_{VIO} is the estimated traveled distance, $[x_i\ y_i]^T$ and $[x_f\ y_f]^T$ are the initial and final 2D position. The results in the room case are shown in tab. 5.1, whereas the ones for the corridor case as reported in tab. 5.2.
In order to find the best camera settings for each algorithm and the best algorithm in each environment a score was defined. First it was computed the average of the

	Jackal		ROVIO		VINS-Fusion		OKVIS	
	δ_d [%]	Δ_p [m]	δ_d [%]	Δ_p [m]	δ_d [%]	Δ_p [m]	δ_d [%]	Δ_p [m]
200 Hz - IR on	1.885	0.041	7.604	0.205	6.926	0.226	1.120	0.217
	0.190	0.099	4.315	0.252	2.512	0.217	0.389	0.044
200 Hz - IR off	0.839	0.061	3.040	0.255	1.131	0.207	0.440	0.052
	0.305	0.073	3.586	0.232	0.425	0.267	1.091	0.121
500 Hz - IR on	1.572	0.075	6.623	0.023	4.033	0.226	1.287	0.049
	0.746	0.191	3.238	0.448	4.759	0.255	0.936	0.116
500 Hz - IR off	0.974	0.076	5.052	0.203	0.369	0.149	0.815	0.069
	1.400	0.064	4.275	0.361	0.368	0.260	2.157	0.047

TABLE 5.1: Room results.

	Jackal		ROVIO		VINS-Fusion		OKVIS	
	δ_d [%]	Δ_p [m]	δ_d [%]	Δ_p [m]	δ_d [%]	Δ_p [m]	δ_d [%]	Δ_p [m]
200 Hz - IR on	0.321	0.400	0.025	0.274	4.653	0.561	5.476	0.267
	0.092	0.482	8.260	0.377	2.171	0.636	5.174	0.411
200 Hz - IR off	0.365	0.411	6.928	0.950	6.638	0.375	5.665	0.216
	0.302	0.033	7.721	0.564	5.996	0.832	5.666	0.303
500 Hz - IR on	0.225	0.506	8.754	0.696	1.129	0.792	4.459	0.315
	0.665	0.712	15.330	0.825	1.137	0.578	4.794	0.265
500 Hz - IR off	0.040	0.607	19.135	0.698	5.944	0.306	5.726	0.233
	0.028	0.462	3.671	1.577	6.498	0.474	5.348	0.223

Table 5.2: Corridor results.

	ROVIO	VINS-Fusion	OKVIS
200 Hz - IR on	**63.0**	81.0	74.0
200 Hz - IR off	127.4	105.5	69.9
500 Hz - IR on	166.9	**69.6**	**63.9**
500 Hz - IR off	194.7	85.9	66.0

Table 5.3: Corridor VIO comparison. Lower is better. The lowest scores for each algorithm are highlighted.

results between the pair of trials for each camera setting (see tab. 5.5 and tab. 5.6), then it was defined a score relative to the worst (highest) value for each metric in one environment:

$$\begin{aligned} s_i^r &= \left(\frac{\mid \delta_{d,i}^r - \delta_{d,max}^r \mid}{\delta_{d,max}^r} + \frac{\mid \Delta_{p,i}^r - \Delta_{p,max}^r \mid}{\Delta_{p,max}^r} \right) \cdot 100, \\ s_i^c &= \left(\frac{\mid \delta_{d,i}^c - \delta_{d,max}^c \mid}{\delta_{d,max}^c} + \frac{\mid \Delta_{p,i}^c - \Delta_{p,max}^c \mid}{\Delta_{p,max}^c} \right) \cdot 100, \end{aligned} \tag{5.1}$$

where s_i^r is the score for case i (camera settings and algorithm) in the averaged results of room environment, $\delta_{d,max}^r$ is the maximum value of δ_d^r in the averaged results of the room environment and the rest is self explanatory.
The result of the procedure (tab. 5.4 and tab. 5.3) shows that there is no algorithm which performs better in both environments (even if OKVIS is pretty close) and that

	ROVIO	VINS-Fusion	OKVIS
200 Hz - IR on	181.2	157.8	58.9
200 Hz - IR off	**142.1**	97.1	**43.6**
500 Hz - IR on	166.1	159.0	47.9
500 Hz - IR off	178.3	**78.7**	45.5

Table 5.4: Room VIO comparison. Lower is better. The lowest scores for each algorithm are highlighted.

	ROVIO		VINS-Fusion		OKVIS	
	δ_d [%]	Δ_p [m]	δ_d [%]	Δ_p [m]	δ_d [%]	Δ_p [m]
200 Hz - IR on	**5.959**	0.229	4.719	0.222	0.755	0.130
200 Hz - IR off	3.313	0.244	0.778	0.237	0.766	0.087
500 Hz - IR on	4.930	0.235	4.396	0.240	1.111	0.083
500 Hz - IR off	4.664	**0.282**	0.368	0.204	1.486	0.058

TABLE 5.5: Room average results. The highest values for each metric are highlighted

	ROVIO		VINS-Fusion		OKVIS	
	δ_d [%]	Δ_p [m]	δ_d [%]	Δ_p [m]	δ_d [%]	Δ_p [m]
200 Hz - IR on	4.142	0.326	3.412	0.598	5.325	0.339
200 Hz - IR off	7.325	0.757	6.317	0.604	5.665	0.260
500 Hz - IR on	**12.042**	0.761	1.133	0.685	4.627	0.290
500 Hz - IR off	11.403	**1.137**	6.221	0.390	5.537	0.228

TABLE 5.6: Corridor average results. The highest values for each metric are highlighted.

the VIO frameworks perform better with the IR projector off the room case and on in the corridor case. This could be caused by lack of textures in the corridor, compared with the room, and the increased number of possible corners increments the estimation accuracy. Instead, in an higher textured environment this is disadvantageous most likely because the higher number of features that can be tracked just for a short time add noise to the estimation.

In conclusion, the camera was calibrated with good results. It was not defined the most accurate VIO framework for indoor environments in the three considered. In fact using the score proposed, in the room environment OKVIS performs better, followed by VINS-Fusion and ROVIO. The results different in the corridor, where ROVIO is better, followed by OKVIS and VINS-Fusion.
More tests are definitely needed, possibly considering a longer path spanning both environments. It would also be interesting to generate and ideal trajectory by interpolating the points between measured positions of the path, using mean velocities. Different parts of the path can be recognized by the linear and angular velocities, since mostly the movements were divided in straight forward or only rotation (aside for small angular correction during the run). This would bound the position drift, the more movement operation changes there are, so it would be better if applied to the corridor path. On the hardware side it would be interesting to see how much is the accuracy boost given by a hardware triggered camera-IMU system (expected from the literature).

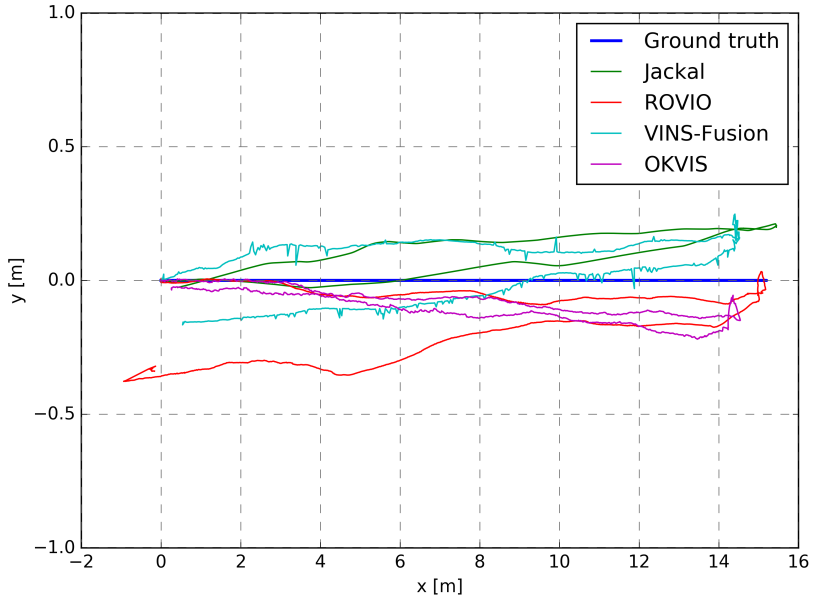


FIGURE 5.3: Trajectory comparison of the best settings for each algorithm in the corridor environment.

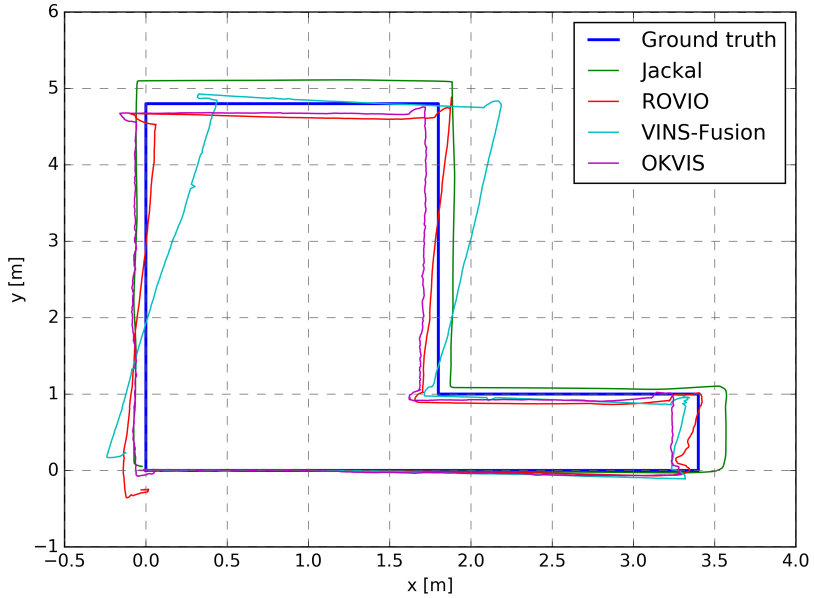


FIGURE 5.4: Trajectory comparison of the best settings for each algorithm in the room environment.

Bibliography

[1] (Jul. 2019), [Online]. Available: https://www.mynteye.com/products/mynt-eye-stereo-camera.

[2] (Aug. 2019), [Online]. Available: http://wiki.ros.org/ROS/Introduction.

[3] J. Salvi, X. Armangué, and J. Batlle, "A comparative review of camera calibrating methods with accuracy evaluation", *Pattern recognition*, vol. 35, no. 7, pp. 1617–1635, 2002.

[4] C. Melchiorri. (Mar. 2012). Ridig body motion – homogeneous transformations, [Online]. Available: http://www-lar.deis.unibo.it/people/cmelchiorri/Files_Robotica/FIR_03_Rbody.pdf.

[5] Y. Morvan, "Acquisition, compression and rendering of depth and texture for multi-view video", 2009.

[6] L. W. Kheng, "Camera models and imaging", *Class lecture, CS4243, National University of Singapore*, 2012.

[7] A. Fusiello, "Elements of geometric computer vision", 2006.

[8] (Oct. 2017). Camera calibration, [Online]. Available: https://docs.opencv.org/3.3.1/dc/dbb/tutorial_py_calibration.html.

[9] X. Liu, Z. Li, K. Zhong, Y. Chao, P. Miraldo, and Y. Shi, "Generic distortion model for metrology under optical microscopes", *Optics and Lasers in Engineering*, vol. 103, pp. 119–126, 2018.

[10] J. Weng, P. Cohen, and M. Herniou, "Camera calibration with distortion models and accuracy evaluation", *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 10, pp. 965–980, 1992.

[11] J. Kannala and S. S. Brandt, "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses", *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 8, pp. 1335–1340, 2006.

[12] V. C. Usenko, N. Demmel, and D. Cremers, "The double sphere camera model", *CoRR*, vol. abs/1807.08957, 2018.

[13] J. Wilm. (Nov. 2018). Calibration best practices, [Online]. Available: https://calib.io/blogs/knowledge-base/calibration-best-practices.

[14] P. Galeone. (Mar. 2018). Camera calibration guidelines, [Online]. Available: https://pgaleone.eu/computer-vision/2018/03/04/camera-calibration-guidelines/.

[15] A. Martos. (Jan. 2017). Which pattern should be used for automotive camera calibration, [Online]. Available: https://www.researchgate.net/post/Which_pattern_circle_pattern_or_checkerboard_pattern_should_be_used_for_automotive_camera_calibration_fisheye_wide_webcam.

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

[17] (Jan. 2019). Mynt eye s sdk guide, [Online]. Available: https://github.com/slightech/MYNT-EYE-S-SDK-Guide.

[18] T. Conceição, B. Coltin, T. Smith, A. Symington, and L. Flückiger, "Joint visual and time-of-flight camera calibration for an automatic procedure in space", Jun. 2018.

[19] T. Hinzmann. (Aug. 2018). Calibration targets, [Online]. Available: https://github.com/ethz-asl/kalibr/wiki/calibration-targets.

[20] K. Hata and S. Savarese, "Cs231a course notes 3: Epipolar geometry", 2017. [Online]. Available: https://web.stanford.edu/class/cs231a/course_notes/03-epipolar-geometry.pdf.

[21] N. Navab and C. Unger, "Rectification and disparity", *Computer Aided Medical Procedures in Technical University Munich*, 2010.

[22] P. T. Furgale, *Extensions to the visual odometry pipeline for the exploration of planetary surfaces*. University of Toronto, 2011.

[23] M. O. Aqel, M. H. Marhaban, M. I. Saripan, and N. B. Ismail, "Review of visual odometry: Types, approaches, challenges, and applications", *SpringerPlus*, vol. 5, no. 1, p. 1897, 2016.

[24] S. Poddar, R. Kottath, and V. Karar, "Evolution of visual odometry techniques", *arXiv preprint arXiv:1804.11142*, 2018.

[25] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden, "Pyramid methods in image processing", *RCA engineer*, vol. 29, no. 6, pp. 33–41, 1984.

[26] A. Jepson and D. Fleet, *Lecture notes on image pyramids*, 2005.

[27] C. Lee. (Aug. 2015). Image pyramid, [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/4/43/Image_pyramid.svg.

[28] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection", in *European conference on computer vision*, Springer, 2006, pp. 430–443.

[29] J. Sánchez, N. Monzón, and A. Salgado De La Nuez, "An analysis and implementation of the harris corner detector", *Image Processing On Line*, 2018.

[30] N. Gandhi. (Jun. 2018). Harris corner detection and shi-tomasi corner detection, [Online]. Available: `https://medium.com/pixel-wise/detect-those-corners-aba0f034078b`.

[31] (Nov. 2014). Shi-tomasi corner detector & good features to track, [Online]. Available: `https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_shi_tomasi/py_shi_tomasi.html`.

[32] J. K. Suhr, "Kanade-lucas-tomasi (klt) feature tracker", *Computer Vision (EEE6503)*, pp. 9–18, 2009.

[33] S. Leutenegger, M. Chli, and R. Siegwart, "Brisk: Binary robust invariant scalable keypoints", in *2011 IEEE international conference on computer vision (ICCV)*, Ieee, 2011, pp. 2548–2555.

[34] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct ekf-based approach", in *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, IEEE, 2015, pp. 298–304.

[35] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback", *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1053–1072, 2017.

[36] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds", *Information Fusion*, vol. 14, no. 1, pp. 57–77, 2013.

[37] J. A. Castellanos, J. Neira, and J. D. Tardós, "Limits to the consistency of ekf-based slam", *IFAC Proceedings Volumes*, vol. 37, no. 8, pp. 716–721, 2004.

[38] D. Zaitsev. (Apr. 2017). What is the difference between linear and affine function, [Online]. Available: `https://math.stackexchange.com/questions/275310/what-is-the-difference-between-linear-and-affine-function`.

[39] S. Yang and H. Li, "Application of ekf and ukf in target tracking problem", in *2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, IEEE, vol. 1, 2016, pp. 116–120.

[40] W. Gander, "Algorithms for the qr decomposition", *Res. Rep*, vol. 80, no. 02, pp. 1251–1268, 1980.

[41] G. J. McLachlan and G. Mclachlan, "Mahalanobis distance", *Resonance*, vol. 4, no. 06, 1999.

[42] T. Qin, J. Pan, S. Cao, and S. Shen, "A general optimization-based framework for local odometry estimation with multiple sensors", *arXiv preprint arXiv:1901.03638*, 2019.

[43] T. Qin and S. Shen, "Online temporal calibration for monocular visual-inertial systems", in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 3662–3669.

[44] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator", *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[45] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography", *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[46] C. Hurlin. (2013). Maximum likelihood estimation, [Online]. Available: `https://www.univ-orleans.fr/deg/masters/ESA/CH/Chapter2_MLE.pdf`.

[47] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, "Keyframe-based visual-inertial slam using nonlinear optimization", *Proceedings of Robotis Science and Systems (RSS) 2013*, 2013.

[48] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization", *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[49] (Jun. 2019). Jackal unmanned ground vehicle, [Online]. Available: `https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/`.

[50] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry", in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2018.

[51] N. Valigi. (Oct. 2016). Trajectory toolkit, [Online]. Available: `https://github.com/ethz-asl/trajectory_toolkit`.

[52] (Jan. 2019). Mynt eye sdk, [Online]. Available: `https://slightech.github.io/MYNT-EYE-SDK/index.html`.