11

Inertial Odometry for SLAM

Guoquan (Paul) Huang, Cédric Le Gentil, Teresa Vidal-Calleja,
Davide Scaramuzza, Frank Dellaert, and Luca Carlone

Inertial Measurement Units (IMUs) have become one of the most pervasive sources of odometry for robot simultaneous localization and mapping. An IMU measures the linear acceleration and the rotation rate of the body the sensor is attached to. IMUs are available in a broad range of form factors, costs, and performance levels, from large and accurate optical sensors used on airplanes to small but more noisy micro-electromechanical systems (MEMS) used in smart phones and other consumer devices. The low-SWAP and inexpensive nature of MEMS IMU sensors renders them great candidates as sensors for robotics, where these sensors have been extensively studied with application to SLAM for more than two decades.

In this chapter, we first introduce basic facts about IMUs and describe their measurement model (Section 11.1). Then, we introduce the concept of IMU preintegration (Section 11.2), which allows adding high-rate IMU data into a factor graph optimization framework. Next, we observe that using IMU data introduces extra variables in the optimization (*e.g.,* sensor biases) and discuss observability[1] properties of systems that combine IMUs with exteroceptive sensors, *e.g.,* camera or LiDAR (Section 11.3). Finally, we showcase examples of what's achievable with modern IMU-centric SLAM systems (Section 11.4) and review recent trends (Section 11.5).

11.1 Basics of Inertial Sensing and Navigation

A 6-axis Inertial Measurement Unit (IMU) comprises an *accelerometer*, which measures the linear acceleration of the sensor with respect to an inertial frame, and a *gyroscope*, which measures the angular velocity (or rotation rate) of the sensor.[2] Traditionally studied in aerospace engineering, *inertial navigation systems (INS)* aim at estimating the current state (*e.g.,* pose, velocity) of the platform the

[1] Observability establishes under which conditions the estimation problem is well posed, *i.e.,* whether it is at all possible to compute an estimate close to the ground truth given the measurements.

[2] An IMU typically also includes a *compass* that measures the direction to the magnetic north. This sensor is less used in SLAM applications since in many robotics applications, including in indoor and urban environments, it exhibits large biases. These biases are induced by local magnetic disturbances, which can be caused by, *e.g.,* large metallic structures and electronic devices.

IMU is mounted on, from the initial state and the history of the IMU measurements [177, 1103]. Different INS can be categorized into *strapdown systems*, where the IMU is mounted to the frame of the platform, and *stabilized systems*, where the IMU is mounted on an inner gimbal, multi-gimbal structure, or floating ball, which is designed to maintain its orientation constant with respect to an inertial frame. Most INS in robotics fall into the former category, *i.e.,* they rely on an IMU that measures the local linear acceleration and angular velocity of the sensing platform it is rigidly connected to. In robotics, the term *inertial odometry* is commonly used as a synonym of inertial navigation, to emphasize the odometric nature of the estimate.

Clearly, the odometric estimate produced by an INS drifts over time, so in most applications the estimation also relies on other sensors (*e.g.,* GPS, cameras, LiDARs), in which case one talks about *aided inertial navigation systems (AINS)*. In robotics, it is common to directly specify the combination of sensors used with the IMU. For instance, a system that combines cameras and IMUs to provide 3D motion tracking is called a *visual-inertial odometry*, while a system that also includes loop closures is called a *visual-inertial SLAM* system.

11.1.1 Sensing Principles and Measurement Models

An IMU commonly includes a 3-axis accelerometer and a 3-axis gyroscope, measuring the angular rate and the linear acceleration of the sensor platform. The basic principle underlying gyroscope design is the conservation of angular momentum. On the other hand, an accelerometer uses the inertia of a mass to measure the difference between the kinematic acceleration with respect to the inertial frame and the gravitational acceleration. Different principles can be used for the design of accelerometers, for example, by using a rate gyroscope mounted as a pendulum mass, based on the inertia of a proof mass inside a low-friction case, or based on the difference in vibration of two thin metal tapes suspended inside a case with a proof mass suspended between them.

Measurement Model. We now describe the IMU measurement model, which relates the IMU measurements to the state of the robot and other quantities (*e.g.,* biases) we need to estimate. For simplicity, we assume the sensor frame coincides with the body frame B of the robot, and the world frame W is an inertial frame.[3] The IMU measurements collected at time t, namely $\dot{\mathbf{v}}(t)$ and $\boldsymbol{\omega}(t)$, are typically assumed to be corrupted by additive white Gaussian noise $\boldsymbol{\eta}$ and slowly varying

[3] In aerospace, it is standard practice to distinguish non-inertial navigation frames, *e.g.,* Earth-Centered Earth-Fixed (ECEF) and Local Geodetic Vertical (LGV) frames, from inertial ones, *e.g.,* Earth-Centered Inertial (ECI) [326]. In robotics, this distinction is often de-emphasized in near-Earth small-scale applications where noisy low-cost IMUs are often used and the impact of the Earth rotation is negligible compared to the measurement noise. For this reason, the world frame W, which in robotics is typically chosen to be fixed at a location on Earth, is approximately treated as an inertial frame.

sensor biases $\mathbf{b}$:

$$\dot{\mathbf{v}}(t) = \boldsymbol{R}_{\mathrm{W}}^{\mathrm{B}}(t)^{\mathsf{T}} \left(\dot{\mathbf{v}}^{\mathrm{W}}(t) - \mathbf{g}^{\mathrm{W}}\right) + \mathbf{b}^{a}(t) + \boldsymbol{\eta}^{a}(t), \tag{11.1}$$

$$\boldsymbol{\omega}(t) = \boldsymbol{\omega}_{\mathrm{WB}}^{\mathrm{B}}(t) + \mathbf{b}^{g}(t) + \boldsymbol{\eta}^{g}(t). \tag{11.2}$$

As usual, the superscript B denotes that the corresponding quantity is expressed in the body (IMU) frame B. The pose of the IMU at time t is described by the transformation $\{\boldsymbol{R}_{\mathrm{B}}^{\mathrm{W}}(t), \mathbf{p}^{\mathrm{W}}(t)\}$, which maps a point from sensor frame B to W; $\dot{\mathbf{v}}^{\mathrm{W}}(t) \in \mathbb{R}^3$ is the acceleration of the sensor in the world frame; $\mathbf{g}^{\mathrm{W}}$ is the gravity vector in the world frame. Therefore, the term $\boldsymbol{R}_{\mathrm{W}}^{\mathrm{B}}(t)\left(\dot{\mathbf{v}}^{\mathrm{W}}(t) - \mathbf{g}^{\mathrm{W}}\right)$ is the acceleration experienced by the IMU in the Body/IMU frame. The vector $\boldsymbol{\omega}_{\mathrm{WB}}^{\mathrm{B}}(t) \in \mathbb{R}^3$ is the instantaneous angular velocity of B relative to W expressed in coordinate frame B. The noise terms $\boldsymbol{\eta}^{g}(t)$ and $\boldsymbol{\eta}^{a}(t)$ are assumed to be zero-mean Gaussian random variables, and the to-be-estimated biases $\mathbf{b}^{a}(t)$ and $\mathbf{b}^{g}(t)$ are assumed to follow random walks. Note that here the superscripts for noise and bias vectors do not refer to the frames but the sensors (accelerometer and gyroscope).

Extended Models. While the IMU measurement model (11.1)-(11.2) often suffices in robotics, more sophisticated models that more accurately model the sensing process may be needed, for example, when (re-)calibrating the sensor platform. Due to the imperfection in manufacturing, accelerometers may suffer from misalignment and scale errors, and the model (11.2) can be extended to:

$$\dot{\mathbf{v}}(t) = \boldsymbol{T}_{a}\, \boldsymbol{R}_{\mathrm{W}}^{\mathrm{B}}(t)^{\mathsf{T}} \left(\dot{\mathbf{v}}^{\mathrm{W}}(t) - \mathbf{g}^{\mathrm{W}}\right) + \mathbf{b}^{a}(t) + \boldsymbol{\eta}^{a}(t), \tag{11.3}$$

where $\boldsymbol{T}_a$ is the shape matrix that models both misalignment and scale errors in the accelerometer measurements. Scale errors can be made of static or temperature-related components and can be determined during sensor (intrinsic) calibration. Similarly, the gyroscope measurement model can be extended to capture misalignment and scale errors:

$$\boldsymbol{\omega}(t) = \boldsymbol{T}_{g}\, \boldsymbol{\omega}_{\mathrm{WB}}^{\mathrm{B}}(t) + \mathbf{b}^{g}(t) + \boldsymbol{\eta}^{g}(t), \tag{11.4}$$

where $\boldsymbol{T}_g$ is the shape matrix that models both misalignment and scale errors in the gyroscope measurements. Gyroscope measurements are often influenced by acceleration, a phenomenon called *g-sensitivity*. The magnitude of this influence is considered negligible if it is within the range of the additive white noise $\boldsymbol{\eta}^{g}(t)$, while in some MEMS hardware, it can be more significant and modeled as follows:

$$\boldsymbol{\omega}(t) = \boldsymbol{T}_{g}\, \boldsymbol{\omega}_{\mathrm{WB}}^{\mathrm{B}}(t) + \boldsymbol{T}_{s}\, \boldsymbol{R}_{\mathrm{W}}^{\mathrm{B}}(t)^{\mathsf{T}} \left(\dot{\mathbf{v}}^{\mathrm{W}}(t) - \mathbf{g}^{\mathrm{W}}\right) + \mathbf{b}^{g}(t) + \boldsymbol{\eta}^{g}(t), \tag{11.5}$$

where $\boldsymbol{T}_s$ is the g-sensitivity matrix, which can be estimated during calibration.

11.1.2 Initial Alignment

In SLAM it is customary to set the global coordinate frame to be the starting pose of the trajectory, *i.e.,* set the initial pose $\{\boldsymbol{R}_{\mathrm{B}}^{\mathrm{W}}(0), \mathbf{p}^{\mathrm{W}}(0)\}$ to be the identity

pose. However, in INS, as the IMU measurements involve the gravitational force (*c.f.* (11.1)), we may choose the world frame to be gravity-aligned, thus requiring to align the initial pose with the gravity direction. In other words, since the IMU measurements depend on the gravity direction, the orientation of the robot is no longer an arbitrary choice, and it must be consistent with the gravity direction. Specifically, we need to compute the rotation $\boldsymbol{R}_{\mathrm{B}}^{\mathrm{W}}(0)$ that aligns the body (IMU) frame to the world frame. For simplicity, assume the robot is initially static, *i.e.,* at the beginning of deployment, no specific force is applied to the robot and the commonly-used low-cost MEMS IMU only measures the gravitational force. Clearly, given only the local gravity measurement $\boldsymbol{g}^{\mathrm{B}}$, we cannot recover the rotation along gravity (*i.e.,* yaw), which is thus up to free choice depending on applications. However, we can determine the rotation corresponding to roll and pitch via the following static initialization:

$$\begin{cases} \mathbf{z}_{\mathrm{W}}^{\mathrm{B}} = \frac{\boldsymbol{g}^{\mathrm{B}}}{||\boldsymbol{g}^{\mathrm{B}}||} \\ \mathbf{x}_{\mathrm{W}}^{\mathrm{B}} = \frac{\mathbf{e}_1 - \mathbf{z}_{\mathrm{W}}^{\mathrm{B}} \mathbf{e}_1^\top \mathbf{z}_{\mathrm{W}}^{\mathrm{B}}}{||\mathbf{e}_1 - \mathbf{z}_{\mathrm{W}}^{\mathrm{B}} \mathbf{e}_1^\top \mathbf{z}_{\mathrm{W}}^{\mathrm{B}}||} \\ \mathbf{y}_{\mathrm{W}}^{\mathrm{B}} = \mathbf{z}_{\mathrm{W}}^{\mathrm{B}} \times \mathbf{x}_{\mathrm{W}}^{\mathrm{B}} \end{cases} \Rightarrow \boldsymbol{R}_{\mathrm{W}}^{\mathrm{B}} = \begin{bmatrix} \mathbf{x}_{\mathrm{W}}^{\mathrm{B}} & \mathbf{y}_{\mathrm{W}}^{\mathrm{B}} & \mathbf{z}_{\mathrm{W}}^{\mathrm{B}} \end{bmatrix} \tag{11.6}$$

where we perform the Gram–Schmidt orthonormalization given vectors $\mathbf{e}_1 = [1\,0\,0]^\top$ and $\boldsymbol{g}^{\mathrm{B}}$, and $\times$ is the cross product. Intuitively, the last column of the rotation matrix $\boldsymbol{R}_{\mathrm{W}}^{\mathrm{B}}$, namely $\boldsymbol{z}_{\mathrm{W}}^{\mathrm{B}}$, is the direction of the z-axis of the world frame with respect to the body frame. Since the z-axis of the world frame is aligned with gravity, eq. (11.6) computes $\boldsymbol{z}_{\mathrm{W}}^{\mathrm{B}}$ from the measurement of the body-frame gravity vector $\boldsymbol{g}^{\mathrm{B}}$. Then, the orthonormalization procedure computes orthonormal vectors $\mathbf{x}_{\mathrm{W}}^{\mathrm{B}}$ and $\mathbf{y}_{\mathrm{W}}^{\mathrm{B}}$ to complete the columns of the rotation matrix $\boldsymbol{R}_{\mathrm{W}}^{\mathrm{B}}$ for an arbitrary choice of yaw.

Alignment with High-end IMUs. When using high-end IMUs, the gyroscope is sensitive enough to measure the Earth rotation rate $\boldsymbol{\omega}_{ie}$. In this case, assuming the chosen world frame is an inertial frame (e.g., the Earth-Centered Inertial frame, or ECI [326]), one can use the measurement of the body-frame gravity vector $\boldsymbol{g}^{\mathrm{B}}$ and the Earth rotation rate $\boldsymbol{\omega}_{ie}$ to perform analytical alignment:

$$\begin{cases} \boldsymbol{g}^{\mathrm{B}} = \boldsymbol{R}_{\mathrm{W}}^{\mathrm{B}} \boldsymbol{g}^{\mathrm{W}} \\ \boldsymbol{\omega}_{ie}^{\mathrm{B}} = \boldsymbol{R}_{\mathrm{W}}^{\mathrm{B}} \boldsymbol{\omega}_{ie}^{\mathrm{W}} \\ \boldsymbol{g}^{\mathrm{B}} \times \boldsymbol{\omega}_{ie}^{\mathrm{B}} = \boldsymbol{R}_{\mathrm{W}}^{\mathrm{B}} \left(\boldsymbol{g}^{\mathrm{W}} \times \boldsymbol{\omega}_{ie}^{\mathrm{W}} \right) \end{cases} \Rightarrow \boldsymbol{R}_{\mathrm{B}}^{\mathrm{W}} = \begin{bmatrix} \boldsymbol{g}^{\mathrm{W}\top} \\ \boldsymbol{\omega}_{ie}^{\mathrm{W}\top} \\ (\boldsymbol{g}^{\mathrm{W}} \times \boldsymbol{\omega}_{ie}^{\mathrm{W}})^\top \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{g}^{\mathrm{B}\top} \\ \boldsymbol{\omega}_{ie}^{\mathrm{B}\top} \\ (\boldsymbol{g}^{\mathrm{B}} \times \boldsymbol{\omega}_{ie}^{\mathrm{W}})^\top \end{bmatrix} \tag{11.7}$$

where the resulting rotation matrix $\boldsymbol{R}_{\mathrm{B}}^{\mathrm{W}}$ typically needs to be projected onto SO(3) to mitigate the impact of measurement noise in practice.

11.2 IMU Preintegration and Factor Graphs

In the previous section, we have introduced the IMU measurement model (11.1)-(11.2), which relates the IMU measurements to the state of the robot, and in partic-

ular its pose and velocity, as well as the sensor biases. While in principle we can use these models to derive a Maximum a Posteriori estimator as described in Chapter 1, this leads to impractically large factor graphs: a typical IMU provides measurements at a high-rate (*e.g.,* 200-1000 Hz) and the measurement model would require adding states to the factor graph at each IMU sampling time. The resulting factor graph would quickly become impractical to solve. The more astute reader might observe that a continuous-time formulation of the problem could circumvent the high-rate addition of variables to the factor graph. However, in a continuous-time formulation of inertial navigation, one would still need to add *factors* at high-rate, one for each measurement, again leading to a quickly growing factor graph.

In this chapter, we present the key idea of *IMU preintegration*, which provides a way to avoid adding states or measurements at IMU rate to the factor graph. Intuitively, we can integrate IMU measurements over time to obtain relative motion measurements, and we can add these motion measurements to the factor graph instead. A naive integration of the IMU measurements (reviewed in Section 11.2.1) would still require repeating the integration of the measurements at each iteration of the factor graph solver. IMU *preintegration* avoids this issue by separating terms that depend on the state variables from the measurements. The original idea of preintegration goes back to [717] and has been extended to operate on manifold in [342, 343]; in Section 11.2.2, we closely follow the presentation in [342, 343]; then discuss more advanced preintegration techniques in Section 11.2.3. As usual, we postpone the discussion of recent works on the topic to Section 11.5.

11.2.1 Motion Integration

In this section, we start by inferring the motion of the robot from IMU measurements. For this purpose we introduce the following kinematic model [796, 326]:

$$\dot{\boldsymbol{R}}_{\mathrm{B}}^{\mathrm{W}} = \boldsymbol{R}_{\mathrm{B}}^{\mathrm{W}} \, (\boldsymbol{\omega}_{\mathrm{WB}}^{\mathrm{B}})^{\wedge}, \qquad \dot{\mathbf{v}}^{\mathrm{W}} = \dot{\mathbf{v}}^{\mathrm{W}}, \qquad \dot{\mathbf{p}}^{\mathrm{W}} = \mathbf{v}^{\mathrm{W}}, \tag{11.8}$$

which describes the evolution of the rotation $\boldsymbol{R}_{\mathrm{B}}^{\mathrm{W}}$, translation $\mathbf{p}^{\mathrm{W}}$, and velocity $\mathbf{v}^{\mathrm{W}}$ of the body frame B with respect to the world frame W.

The state at time $t + \Delta t$, where Δt is the IMU sampling period, is obtained by integrating (11.8):

$$\boldsymbol{R}_{\mathrm{B}}^{\mathrm{W}}(t + \Delta t) = \boldsymbol{R}_{\mathrm{B}}^{\mathrm{W}}(t) \, \mathrm{Exp}\left(\int_{t}^{t+\Delta t} \boldsymbol{\omega}_{\mathrm{WB}}^{\mathrm{B}}(\tau) d\tau \right) \tag{11.9}$$

$$\mathbf{v}^{\mathrm{W}}(t + \Delta t) = \mathbf{v}^{\mathrm{W}}(t) + \int_{t}^{t+\Delta t} \dot{\mathbf{v}}^{\mathrm{W}}(\tau) d\tau$$

$$\mathbf{p}^{\mathrm{W}}(t + \Delta t) = \mathbf{p}^{\mathrm{W}}(t) + \int_{t}^{t+\Delta t} \mathbf{v}^{\mathrm{W}}(\tau) d\tau \tag{11.10}$$

where in the first equation we assumed that the direction of the angular velocity

$\boldsymbol{\omega}^{\mathrm{B}}_{\mathrm{WB}}$ does not change in the interval $[t, t+\Delta t]$.[4] Further assuming that $\dot{\mathbf{v}}^{\mathrm{W}}$ and $\boldsymbol{\omega}^{\mathrm{B}}_{\mathrm{WB}}$ remain constant in the time interval $[t, t+\Delta t]$, we can write:

$$\begin{aligned}
\boldsymbol{R}^{\mathrm{W}}_{\mathrm{B}}(t+\Delta t) &= \boldsymbol{R}^{\mathrm{W}}_{\mathrm{B}}(t)\,\mathrm{Exp}\left(\boldsymbol{\omega}^{\mathrm{B}}_{\mathrm{WB}}(t)\Delta t\right)\\
\mathbf{v}^{\mathrm{W}}(t+\Delta t) &= \mathbf{v}^{\mathrm{W}}(t) + \dot{\mathbf{v}}^{\mathrm{W}}(t)\Delta t\\
\mathbf{p}^{\mathrm{W}}(t+\Delta t) &= \mathbf{p}^{\mathrm{W}}(t) + \mathbf{v}^{\mathrm{W}}(t)\Delta t + \frac{1}{2}\dot{\mathbf{v}}^{\mathrm{W}}(t)\Delta t^2.
\end{aligned} \tag{11.11}$$

More generally, eq. (11.11) can be understood as applying Euler integration to numerically solve the integrals in (11.9). Using Eqs. (11.1)-(11.2), we can write $\dot{\mathbf{v}}^{\mathrm{W}}$ and $\boldsymbol{\omega}^{\mathrm{B}}_{\mathrm{WB}}$ as functions of the IMU measurements, hence (11.11) becomes

$$\begin{aligned}
\boldsymbol{R}(t+\Delta t) =&\ \boldsymbol{R}(t)\,\mathrm{Exp}\left(\left(\tilde{\boldsymbol{\omega}}(t) - \mathbf{b}^g(t) - \boldsymbol{\eta}^{gd}(t)\right)\Delta t\right)\\
\mathbf{v}(t+\Delta t) =&\ \mathbf{v}(t) + \mathbf{g}\Delta t + \boldsymbol{R}(t)\left(\tilde{\mathbf{v}}(t) - \mathbf{b}^a(t) - \boldsymbol{\eta}^{ad}(t)\right)\Delta t\\
\mathbf{p}(t+\Delta t) =&\ \mathbf{p}(t) + \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{g}\Delta t^2 + \frac{1}{2}\boldsymbol{R}(t)\left(\tilde{\mathbf{v}}(t) - \mathbf{b}^a(t) - \boldsymbol{\eta}^{ad}(t)\right)\Delta t^2,
\end{aligned} \tag{11.12}$$

where we dropped the coordinate frame subscripts for readability (the notation should be unambiguous from now on). This numeric integration of the velocity and position assumes a constant orientation $\boldsymbol{R}(t)$ for the time of integration between two measurements, which is not an exact solution of the differential equation (11.8) for measurements with non-zero rotation rate. In practice, the use of a high-rate IMU mitigates the effects of this approximation. We adopt the integration scheme (11.12) as it is simple and amenable for modeling and uncertainty propagation, and then discuss more advanced integration techniques in Section 11.2.3. The covariance of the discrete-time noise $\boldsymbol{\eta}^{gd}$ is a function of the sampling rate and relates to the continuous-time noise $\boldsymbol{\eta}^{g}$ via $\mathrm{Cov}(\boldsymbol{\eta}^{gd}(t)) = \frac{1}{\Delta t}\mathrm{Cov}(\boldsymbol{\eta}^{g}(t))$. The same relation holds for $\boldsymbol{\eta}^{ad}$ (*cf.*, [240, Appendix]).

While Eq. (11.12) could be readily seen as a probabilistic constraint in a factor graph, it would require including states in the factor graph at high rate. Intuitively, Eq. (11.12) relates states at time t and $t+\Delta t$, where Δt is the sampling period of the IMU, hence we would have to add new states in the estimation at every new IMU measurement [511].

We can try to avoid this issue by integrating over longer time intervals. In particular, if we assume that we already have a factor graph modeling other sensor measurements in our problem (*e.g.,* the vision measurements in Chapter 7), we can use the expression (11.12) and integrate IMU measurements between two temporally consecutive states in our factor graph. We are going to refer to these states as "keyframe states".[5] Iterating the IMU integration (11.12) for all Δt intervals

[4] We provide a more general expression for the rotation integration in eq. (11.35) below.

[5] We use this terminology, since in many applications involving IMUs and cameras, the states in the factor graph are added at a subset of the camera frames, namely the *keyframes*. However, this term

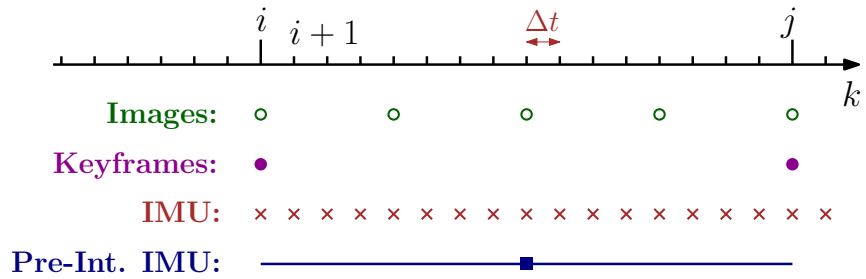

Figure 11.1 Different rates for IMU and camera. From [343] (©2016 IEEE).

between two consecutive keyframes at times t_i and t_j (*c.f.*, Fig. 11.1), we get:[6]

$$\begin{aligned}
\boldsymbol{R}_j &= \boldsymbol{R}_i \prod_{k=i}^{j-1} \mathrm{Exp}\left(\left(\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_k^g - \boldsymbol{\eta}_k^{gd}\right)\Delta t\right), \\
\mathbf{v}_j &= \mathbf{v}_i + \mathbf{g}\Delta t_{ij} + \sum_{k=i}^{j-1} \boldsymbol{R}_k\left(\tilde{\mathbf{v}}_k - \mathbf{b}_k^a - \boldsymbol{\eta}_k^{ad}\right)\Delta t \\
\mathbf{p}_j &= \mathbf{p}_i + \sum_{k=i}^{j-1}\left[\mathbf{v}_k\Delta t + \frac{1}{2}\mathbf{g}\Delta t^2 + \frac{1}{2}\boldsymbol{R}_k\left(\tilde{\mathbf{v}}_k - \mathbf{b}_k^a - \boldsymbol{\eta}_k^{ad}\right)\Delta t^2\right]
\end{aligned} \tag{11.13}$$

where we introduced the shorthands $\Delta t_{ij} \doteq \sum_{k=i}^{j-1}\Delta t$ and $(\cdot)_i \doteq (\cdot)(t_i)$ for readability. While Eq. (11.13) already provides an estimate of the motion between time t_i and t_j, it has the drawback that the integration in (11.13) has to be repeated whenever the linearization point at time t_i changes [652] (*e.g.*, at each iteration of a Gauss-Newton solver). For instance, a change in the rotation $\boldsymbol{R}_i$ implies a change in all future rotations $\boldsymbol{R}_k$, $k = i, \ldots, j-1$, and makes necessary to re-evaluate summations and products in (11.13).

11.2.2 IMU Preintegration on Manifold

Here we show that a small manipulation of the motion integration results (11.13) allows computing relative measurements between states at time t_i and t_j that do not need to be recomputed when the linearization point changes. The key insight is to express measurements in a local frame (such that they do not change when the global state estimate of the robot changes) and isolating the contribution of gravity (which again carries information about the global frame). This process

is used without loss of generality here, and one can decide to instantiate keyframe states arbitrarily (*e.g.*, at every camera frame, LiDAR scans, every "n" IMU measurements, etc).

[6] For simplicity, we assume that the IMU is synchronized with the other sensors, and IMU measurements are sampled at time t_i and t_j. In practice, one can interpolate measurements to approximate the case where IMU measurements are exactly sampled at time t_i and t_j; see Section 11.4.3 for further discussion about temporal synchronization.

leads to computing the so called *preintegrated IMU measurements*, which constrain the motion between consecutive states in the factor graph.

Towards this goal, we define the following relative motion increments that are independent of the pose and velocity at t_i:

$$\begin{aligned}
\Delta\boldsymbol{R}_{ij} &\doteq \boldsymbol{R}_i^\mathsf{T}\boldsymbol{R}_j = \prod_{k=i}^{j-1} \mathrm{Exp}\left(\left(\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_k^g - \boldsymbol{\eta}_k^{gd}\right)\Delta t\right) \\
\Delta\mathbf{v}_{ij} &\doteq \boldsymbol{R}_i^\mathsf{T}\left(\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{ij}\right) = \sum_{k=i}^{j-1} \Delta\boldsymbol{R}_{ik}\left(\tilde{\mathbf{v}}_k - \mathbf{b}_k^a - \boldsymbol{\eta}_k^{ad}\right)\Delta t \\
\Delta\mathbf{p}_{ij} &\doteq \boldsymbol{R}_i^\mathsf{T}\left(\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i\Delta t_{ij} - \tfrac{1}{2}\mathbf{g}\Delta t_{ij}^2\right) \\
&= \sum_{k=i}^{j-1}\left[\Delta\mathbf{v}_{ik}\Delta t + \frac{1}{2}\Delta\boldsymbol{R}_{ik}\left(\tilde{\mathbf{v}}_k - \mathbf{b}_k^a - \boldsymbol{\eta}_k^{ad}\right)\Delta t^2\right],
\end{aligned} \tag{11.14}$$

where $\Delta\boldsymbol{R}_{ik} \doteq \boldsymbol{R}_i^\mathsf{T}\boldsymbol{R}_k$ and $\Delta\mathbf{v}_{ik} \doteq \boldsymbol{R}_i^\mathsf{T}\left(\mathbf{v}_k - \mathbf{v}_i - \mathbf{g}\Delta t_{ik}\right)$. We highlight that, in contrast to the "delta" rotation $\Delta\boldsymbol{R}_{ij}$, neither $\Delta\mathbf{v}_{ij}$ nor $\Delta\mathbf{p}_{ij}$ correspond to the true *physical* change in velocity and position but are defined in a way that make the right-hand side of (11.14) independent from the state at time i as well as gravitational effects. Indeed, we will be able to compute the right-hand side of (11.14) directly from the inertial measurements between the two keyframes.

Unfortunately, summations and products in (11.14) are still function of the bias estimate. We tackle this problem in two steps. In Section 11.2.2.1, we assume $\mathbf{b}_i$ is known; then, in Section 11.2.2.3 we show how to avoid repeating the integration when the bias estimate changes. In both cases, we assume the biases remain constant between times t_i and t_j:

$$\mathbf{b}_i^g = \mathbf{b}_{i+1}^g = \ldots = \mathbf{b}_{j-1}^g, \quad \mathbf{b}_i^a = \mathbf{b}_{i+1}^a = \ldots = \mathbf{b}_{j-1}^a. \tag{11.15}$$

11.2.2.1 Preintegrated IMU Measurements

Equation (11.14) relates the states of keyframes i and j (left-hand side) to the measurements (right-hand side). In this sense, it can be already understood as a measurement model. Unfortunately, it has a fairly intricate dependence on the measurement noise and this complicates a direct application of MAP estimation; intuitively, the MAP estimator requires to clearly define the densities (and their log-likelihood) of the measurements. In this section we manipulate (11.14) so to make easier the derivation of the measurement log-likelihood. More concretely, we isolate the noise terms of the individual inertial measurements in (11.14). As discussed above, within this section assume that the bias at time t_i is known.

Let us start with the rotation increment $\Delta\boldsymbol{R}_{ij}$ in (11.14). Towards this goal, we

use the following properties of the exponential map for SO(3):

$$\mathrm{Exp}(\boldsymbol{\phi}+\delta\boldsymbol{\phi}) \approx \mathrm{Exp}(\boldsymbol{\phi})\,\mathrm{Exp}(\mathsf{J}_r(\boldsymbol{\phi})\delta\boldsymbol{\phi}), \tag{11.16}$$

$$\mathrm{Exp}(\boldsymbol{\phi})\;\boldsymbol{R} = \boldsymbol{R}\,\mathrm{Exp}(\boldsymbol{R}^{\mathsf{T}}\boldsymbol{\phi}). \tag{11.17}$$

where the first relations is a first-order approximation of the exponential of a sum, and the second can be derived from the group's adjoint representation.

Using (11.16) and (11.17), we rearrange the terms in the expression of $\Delta\boldsymbol{R}_{ij}$ in (11.14), by "moving" the noise to the end:

$$\begin{aligned}
\Delta\boldsymbol{R}_{ij} &\overset{\text{eq.(11.16)}}{\simeq} \prod_{k=i}^{j-1}\left[\mathrm{Exp}\left((\tilde{\boldsymbol{\omega}}_k-\mathbf{b}_i^g)\,\Delta t\right)\mathrm{Exp}\left(-\mathsf{J}_r^k\;\boldsymbol{\eta}_k^{gd}\;\Delta t\right)\right] \\
&\overset{\text{eq.(11.17)}}{=} \Delta\tilde{\mathbb{R}}_{ij}\prod_{k=i}^{j-1}\mathrm{Exp}\left(-\Delta\tilde{\mathbb{R}}_{k+1j}^{\mathsf{T}}\;\mathsf{J}_r^k\;\boldsymbol{\eta}_k^{gd}\;\Delta t\right) \\
&\doteq \Delta\tilde{\mathbb{R}}_{ij}\mathrm{Exp}\left(-\delta\boldsymbol{\phi}_{ij}\right)
\end{aligned} \tag{11.18}$$

with $\mathsf{J}_r^k \doteq \mathsf{J}_r^k((\tilde{\boldsymbol{\omega}}_k-\mathbf{b}_i^g)\Delta t)$. In the last line of (11.18), we defined the *preintegrated rotation measurement* $\Delta\tilde{\mathbb{R}}_{ij} \doteq \prod_{k=i}^{j-1}\mathrm{Exp}\left((\tilde{\boldsymbol{\omega}}_k-\mathbf{b}_i^g)\,\Delta t\right)$, and its noise $\delta\boldsymbol{\phi}_{ij}$, which will be further analysed in the next section.

Similarly, we can manipulate the velocity and position equations in (11.14) by using the following relations:

$$\exp(\boldsymbol{\phi}^\wedge) \approx \mathbf{I}+\boldsymbol{\phi}^\wedge, \tag{11.19}$$

$$\mathbf{a}^\wedge\,\mathbf{b} = -\mathbf{b}^\wedge\,\mathbf{a}, \quad \forall\,\mathbf{a},\mathbf{b}\in\mathbb{R}^3, \tag{11.20}$$

where the first relation is a first-order approximation of the exponential map at the origin, while the second is a property of the wedge operator of a vector.

Substituting (11.18) back into the expression of $\Delta\mathbf{v}_{ij}$ in (11.14), using the first-order approximation (11.19) for $\mathrm{Exp}\left(-\delta\boldsymbol{\phi}_{ij}\right)$, and dropping higher-order noise terms, we obtain:

$$\begin{aligned}
\Delta\mathbf{v}_{ij} &\overset{\text{eq.(11.19)}}{\simeq} \sum_{k=i}^{j-1}\Delta\tilde{\mathbb{R}}_{ik}(\mathbf{I}-\delta\boldsymbol{\phi}_{ik}^\wedge)\left(\tilde{\dot{\mathbf{v}}}_k-\mathbf{b}_i^a\right)\Delta t-\Delta\tilde{\mathbb{R}}_{ik}\boldsymbol{\eta}_k^{ad}\Delta t \\
&\overset{\text{eq.(11.20)}}{=} \Delta\tilde{\mathbf{v}}_{ij}+\sum_{k=i}^{j-1}\left[\Delta\tilde{\mathbb{R}}_{ik}\left(\tilde{\dot{\mathbf{v}}}_k-\mathbf{b}_i^a\right)^\wedge\delta\boldsymbol{\phi}_{ik}\Delta t-\Delta\tilde{\mathbb{R}}_{ik}\boldsymbol{\eta}_k^{ad}\Delta t\right] \\
&\doteq \Delta\tilde{\mathbf{v}}_{ij}-\delta\mathbf{v}_{ij}
\end{aligned} \tag{11.21}$$

where we defined the *preintegrated velocity measurement* $\Delta\tilde{\mathbf{v}}_{ij} \doteq \sum_{k=i}^{j-1}\Delta\tilde{\mathbb{R}}_{ik}\left(\tilde{\dot{\mathbf{v}}}_k-\mathbf{b}_i^a\right)\Delta t$ and its noise $\delta\mathbf{v}_{ij}$.

Similarly, substituting (11.18) and (11.21) in the expression of $\Delta\mathbf{p}_{ij}$ in (11.14),

and using the first-order approximation (11.19), we obtain:

$$\begin{aligned}\Delta\mathbf{p}_{ij} &\overset{\text{eq.(11.19)}}{\simeq} \sum_{k=i}^{j-1}\Big[(\Delta\tilde{\mathbf{v}}_{ik}-\delta\mathbf{v}_{ik})\Delta t+\frac{1}{2}\Delta\tilde{\mathbb{R}}_{ik}(\mathbf{I}-\delta\boldsymbol{\phi}_{ik}^{\wedge})\left(\tilde{\mathbf{v}}_k-\mathbf{b}_i^a\right)\Delta t^2 \\ &\qquad -\frac{1}{2}\Delta\tilde{\mathbb{R}}_{ik}\boldsymbol{\eta}_k^{ad}\Delta t^2\Big] \\ &\overset{\text{eq.(11.20)}}{=} \Delta\tilde{\mathbf{p}}_{ij}+\sum_{k=i}^{j-1}\Big[-\delta\mathbf{v}_{ik}\Delta t+\frac{1}{2}\Delta\tilde{\mathbb{R}}_{ik}\left(\tilde{\mathbf{v}}_k-\mathbf{b}_i^a\right)^{\wedge}\delta\boldsymbol{\phi}_{ik}\Delta t^2 \\ &\qquad -\frac{1}{2}\Delta\tilde{\mathbb{R}}_{ik}\boldsymbol{\eta}_k^{ad}\Delta t^2\Big] \\ &\doteq \Delta\tilde{\mathbf{p}}_{ij}-\delta\mathbf{p}_{ij}, \end{aligned} \tag{11.22}$$

where we defined the *preintegrated position measurement* $\Delta\tilde{\mathbf{p}}_{ij}$ and its noise $\delta\mathbf{p}_{ij}$.

Substituting the expressions (11.18), (11.21), (11.22) back in the original definition of $\Delta\boldsymbol{R}_{ij}, \Delta\mathbf{v}_{ij}, \Delta\mathbf{p}_{ij}$ in (11.14), we finally get our *preintegrated measurement model* (remember $\text{Exp}\,(-\delta\boldsymbol{\phi}_{ij})^{\mathsf{T}} = \text{Exp}\,(\delta\boldsymbol{\phi}_{ij})$):

$$\begin{aligned}\Delta\tilde{\mathbb{R}}_{ij} &= \boldsymbol{R}_i^{\mathsf{T}}\boldsymbol{R}_j\text{Exp}\,(\delta\boldsymbol{\phi}_{ij}) \\ \Delta\tilde{\mathbf{v}}_{ij} &= \boldsymbol{R}_i^{\mathsf{T}}\left(\mathbf{v}_j-\mathbf{v}_i-\mathbf{g}\Delta t_{ij}\right)+\delta\mathbf{v}_{ij} \\ \Delta\tilde{\mathbf{p}}_{ij} &= \boldsymbol{R}_i^{\mathsf{T}}\left(\mathbf{p}_j-\mathbf{p}_i-\mathbf{v}_i\Delta t_{ij}-\frac{1}{2}\mathbf{g}\Delta t_{ij}^2\right)+\delta\mathbf{p}_{ij}\end{aligned} \tag{11.23}$$

where our compound measurements are written as a function of the (to-be-estimated) state "plus" a random noise, described by the random vector $[\delta\boldsymbol{\phi}_{ij}^{\mathsf{T}}, \delta\mathbf{v}_{ij}^{\mathsf{T}}, \delta\mathbf{p}_{ij}^{\mathsf{T}}]^{\mathsf{T}}$.

In summary, in this section we manipulated the measurement model (11.14) and rewrote it as (11.23). The advantage of the measurements in eq. (11.23) is that, for a suitable distribution of the noise, they can be used directly to instantiate factors between states at time t_i and t_j in our factor graph. The nature of the noise terms is discussed in the following section.

11.2.2.2 Noise Propagation

In this section we derive the statistics of the noise vector $[\delta\boldsymbol{\phi}_{ij}^{\mathsf{T}}, \delta\mathbf{v}_{ij}^{\mathsf{T}}, \delta\mathbf{p}_{ij}^{\mathsf{T}}]^{\mathsf{T}}$. While we already observed that it is convenient to approximate the noise vector to be zero-mean Normally distributed, it is of paramount importance to accurately model the noise covariance. In this section, we therefore provide a derivation of the covariance $\boldsymbol{\Sigma}_{ij}$ of the preintegrated measurements:

$$\boldsymbol{\eta}_{ij}^{\Delta} \doteq [\delta\boldsymbol{\phi}_{ij}^{\mathsf{T}}, \delta\mathbf{v}_{ij}^{\mathsf{T}}, \delta\mathbf{p}_{ij}^{\mathsf{T}}]^{\mathsf{T}} \sim \mathcal{N}(\mathbf{0}_{9\times 1}, \boldsymbol{\Sigma}_{ij}). \tag{11.24}$$

We first consider the preintegrated rotation noise $\delta\boldsymbol{\phi}_{ij}$. Recall from (11.18) that

$$\text{Exp}\,(-\delta\boldsymbol{\phi}_{ij}) \doteq \textstyle\prod_{k=i}^{j-1}\text{Exp}\left(-\Delta\tilde{\mathbb{R}}_{k+1j}^{\mathsf{T}}\mathtt{J}_r^k\,\boldsymbol{\eta}_k^{gd}\,\Delta t\right). \tag{11.25}$$

Taking the Log on both sides and changing signs, we get:

$$\delta\boldsymbol{\phi}_{ij} = -\text{Log}\left(\textstyle\prod_{k=i}^{j-1}\text{Exp}\left(-\Delta\tilde{\mathbb{R}}_{k+1j}^{\mathsf{T}}\mathtt{J}_r^k\,\boldsymbol{\eta}_k^{gd}\,\Delta t\right)\right). \tag{11.26}$$

Next, we use the following first-order approximation holds for SO(3) logarithm:

$$\mathrm{Log}\big(\,\mathrm{Exp}(\boldsymbol{\phi})\,\mathrm{Exp}(\delta\boldsymbol{\phi})\,\big) \approx \boldsymbol{\phi} + \mathtt{J}_r^{-1}(\boldsymbol{\phi})\delta\boldsymbol{\phi}. \tag{11.27}$$

where $\mathtt{J}_r^{-1}(\boldsymbol{\phi})$ is the inverse of the right Jacobian. Repeated application of (11.27) (recall that $\boldsymbol{\eta}_k^{gd}$ as well as $\delta\boldsymbol{\phi}_{ij}$ are small rotation noises, hence the right Jacobians are close to the identity) produces:

$$\delta\boldsymbol{\phi}_{ij} \simeq \textstyle\sum_{k=i}^{j-1} \Delta\tilde{\mathbb{R}}_{k+1j}^{\mathsf{T}}\, \mathtt{J}_r^k\, \boldsymbol{\eta}_k^{gd}\, \Delta t \tag{11.28}$$

Up to first order, the noise $\delta\boldsymbol{\phi}_{ij}$ is zero-mean and Gaussian, as it is a linear combination of zero-mean noise terms $\boldsymbol{\eta}_k^{gd}$.

Dealing with the noise terms $\delta\mathbf{v}_{ij}$ and $\delta\mathbf{p}_{ij}$ is now easy: these are linear combinations of the acceleration noise $\boldsymbol{\eta}_k^{ad}$ and the preintegrated rotation noise $\delta\boldsymbol{\phi}_{ij}$, hence they are also zero-mean and Gaussian. Simple manipulation leads to:

$$\begin{aligned}
\delta\mathbf{v}_{ij} &\simeq \sum_{k=i}^{j-1} \Big[-\Delta\tilde{\mathbb{R}}_{ik}\left(\tilde{\mathbf{v}}_k - \mathbf{b}_i^a\right)^\wedge \delta\boldsymbol{\phi}_{ik}\Delta t + \Delta\tilde{\mathbb{R}}_{ik}\boldsymbol{\eta}_k^{ad}\Delta t\Big] \\
\delta\mathbf{p}_{ij} &\simeq \sum_{k=i}^{j-1} \Big[\delta\mathbf{v}_{ik}\Delta t - \frac{1}{2}\Delta\tilde{\mathbb{R}}_{ik}\left(\tilde{\mathbf{v}}_k - \mathbf{b}_i^a\right)^\wedge \delta\boldsymbol{\phi}_{ik}\Delta t^2 + \frac{1}{2}\Delta\tilde{\mathbb{R}}_{ik}\boldsymbol{\eta}_k^{ad}\Delta t^2\Big]
\end{aligned} \tag{11.29}$$

where the relations are valid up to the first order.

Eqs. (11.28)-(11.29) express the preintegrated noise $\boldsymbol{\eta}_{ij}^{\Delta}$ as a linear function of the IMU measurement noise $\boldsymbol{\eta}_k^d \doteq [\boldsymbol{\eta}_k^{gd}, \boldsymbol{\eta}_k^{ad}]$, $k = 1, \ldots, j-1$. Therefore, from the knowledge of the covariance of $\boldsymbol{\eta}_k^d$ (given in the IMU specifications), we can compute the covariance of $\boldsymbol{\eta}_{ij}^{\Delta}$, namely $\boldsymbol{\Sigma}_{ij}$, by a simple linear propagation.

An extended derivation of the noise propagation can be found in [343], which also provides an iterative expression to compute the covariance by incrementally adding new measurements as they are collected. The iterative computation leads to simpler expressions and is more amenable for online inference.

11.2.2.3 Incorporating Bias Updates

In the previous section, we assumed that the bias $\{\bar{\mathbf{b}}_i^a, \bar{\mathbf{b}}_i^g\}$ that is used during preintegration between $k = i$ and $k = j$ is correct and does not change. However, more likely, the bias estimate changes by a small amount $\delta\mathbf{b}$ during optimization. One solution would be to recompute the delta measurements when the bias changes; however, that is computationally expensive. Instead, given a bias update $\mathbf{b} \leftarrow \bar{\mathbf{b}} + \delta\mathbf{b}$, we can update the delta measurements using a first-order expansion:

$$\begin{aligned}
\Delta\tilde{\mathbb{R}}_{ij}(\mathbf{b}_i^g) &\simeq \Delta\tilde{\mathbb{R}}_{ij}(\bar{\mathbf{b}}_i^g)\,\mathrm{Exp}\Big(\tfrac{\partial\Delta\bar{\mathbb{R}}_{ij}}{\partial\mathbf{b}^g}\delta\mathbf{b}^g\Big) \\
\Delta\tilde{\mathbf{v}}_{ij}(\mathbf{b}_i^g, \mathbf{b}_i^a) &\simeq \Delta\tilde{\mathbf{v}}_{ij}(\bar{\mathbf{b}}_i^g, \bar{\mathbf{b}}_i^a) + \tfrac{\partial\Delta\bar{\mathbf{v}}_{ij}}{\partial\mathbf{b}^g}\delta\mathbf{b}_i^g + \tfrac{\partial\Delta\bar{\mathbf{v}}_{ij}}{\partial\mathbf{b}^a}\delta\mathbf{b}_i^a \\
\Delta\tilde{\mathbf{p}}_{ij}(\mathbf{b}_i^g, \mathbf{b}_i^a) &\simeq \Delta\tilde{\mathbf{p}}_{ij}(\bar{\mathbf{b}}_i^g, \bar{\mathbf{b}}_i^a) + \tfrac{\partial\Delta\bar{\mathbf{p}}_{ij}}{\partial\mathbf{b}^g}\delta\mathbf{b}_i^g + \tfrac{\partial\Delta\bar{\mathbf{p}}_{ij}}{\partial\mathbf{b}^a}\delta\mathbf{b}_i^a
\end{aligned} \tag{11.30}$$

This is similar to the bias correction in [717] but operates directly on SO(3). The Jacobians $\{\frac{\partial \Delta\bar{\mathbb{R}}_{ij}}{\partial \mathbf{b}^g}, \frac{\partial \Delta\bar{\mathbf{v}}_{ij}}{\partial \mathbf{b}^g}, \ldots\}$ (computed at $\bar{\mathbf{b}}_i$, the bias estimate at integration time) describe how the measurements change due to a change in the bias estimate. The Jacobians remain constant and can be precomputed during the preintegration. The derivation of the Jacobians is very similar to the one we used in Section 11.2.2.1 to express the measurements as a large value *plus* a small perturbation and is given in [343].

11.2.2.4 Preintegrated IMU Factors and Bias Models

Given the preintegrated measurement model in (11.23) and since measurement noise is zero-mean and Gaussian (with covariance $\boldsymbol{\Sigma}_{ij}$) up to first order (11.24), it is now easy to write the residual errors $\mathbf{r}_{\mathcal{I}_{ij}} \doteq [\mathbf{r}^\mathsf{T}_{\Delta \boldsymbol{R}_{ij}}, \mathbf{r}^\mathsf{T}_{\Delta \mathbf{v}_{ij}}, \mathbf{r}^\mathsf{T}_{\Delta \mathbf{p}_{ij}}]^\mathsf{T} \in \mathbb{R}^9$, which will appear in the factor graph optimization:

$$
\begin{aligned}
\mathbf{r}_{\Delta \boldsymbol{R}_{ij}} &\doteq \operatorname{Log}\left(\left(\Delta\tilde{\mathbb{R}}_{ij}(\bar{\mathbf{b}}_i^g)\operatorname{Exp}\left(\tfrac{\partial \Delta\bar{\mathbb{R}}_{ij}}{\partial \mathbf{b}^g}\delta\mathbf{b}^g\right)\right)^\mathsf{T} \boldsymbol{R}_i^\mathsf{T}\boldsymbol{R}_j\right) \\
\mathbf{r}_{\Delta \mathbf{v}_{ij}} &\doteq \boldsymbol{R}_i^\mathsf{T}\left(\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{ij}\right) \\
&\quad - \left[\Delta\tilde{\mathbf{v}}_{ij}(\bar{\mathbf{b}}_i^g, \bar{\mathbf{b}}_i^a) + \tfrac{\partial \Delta\bar{\mathbf{v}}_{ij}}{\partial \mathbf{b}^g}\delta\mathbf{b}^g + \tfrac{\partial \Delta\bar{\mathbf{v}}_{ij}}{\partial \mathbf{b}^a}\delta\mathbf{b}^a\right] \\
\mathbf{r}_{\Delta \mathbf{p}_{ij}} &\doteq \boldsymbol{R}_i^\mathsf{T}\left(\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i\Delta t_{ij} - \tfrac{1}{2}\mathbf{g}\Delta t_{ij}^2\right) \\
&\quad - \left[\Delta\tilde{\mathbf{p}}_{ij}(\bar{\mathbf{b}}_i^g, \bar{\mathbf{b}}_i^a) + \tfrac{\partial \Delta^{\bar{\mathsf{T}}}{}_{ij}}{\partial \mathbf{b}^g}\delta\mathbf{b}^g + \tfrac{\partial \Delta^{\bar{\mathsf{T}}}{}_{ij}}{\partial \mathbf{b}_a}\delta\mathbf{b}^a\right],
\end{aligned}
\tag{11.31}
$$

in which we also included the bias updates of Eq. (11.30). These terms can be readily added to the factor graph by adding the term $\left\|\mathbf{r}_{\mathcal{I}_{ij}}\right\|^2_{\boldsymbol{\Sigma}_{ij}}$ to the objective of the minimization.

When presenting the IMU model (11.1)-(11.2), we said that biases are slowly time-varying quantities. Hence, we model them with a "Brownian motion", *i.e.*, integrated white noise:

$$
\dot{\mathbf{b}}^g(t) = \boldsymbol{\eta}^{bg}, \qquad \dot{\mathbf{b}}^a(t) = \boldsymbol{\eta}^{ba}. \tag{11.32}
$$

Integrating (11.32) over the time interval $[t_i, t_j]$ between two consecutive keyframes i and j we get:

$$
\mathbf{b}_j^g = \mathbf{b}_i^g + \boldsymbol{\eta}^{bgd}, \qquad \mathbf{b}_j^a = \mathbf{b}_i^a + \boldsymbol{\eta}^{bad}, \tag{11.33}
$$

where, as done before, we use the shorthand $\mathbf{b}_i^g \doteq \mathbf{b}^g(t_i)$, and we define the discrete noises $\boldsymbol{\eta}^{bgd}$ and $\boldsymbol{\eta}^{bad}$, which have zero mean and covariance $\boldsymbol{\Sigma}^{bgd} \doteq \Delta t_{ij}\operatorname{Cov}(\boldsymbol{\eta}^{bg})$ and $\boldsymbol{\Sigma}^{bad} \doteq \Delta t_{ij}\operatorname{Cov}(\boldsymbol{\eta}^{ba})$, respectively (*cf.* [240, Appendix]).

The model (11.33) can be readily included in our factor graph, as a further additive term in the objective function, for all consecutive keyframes:

$$
\|\mathbf{r}_{\mathbf{b}_{ij}}\|^2 \doteq \|\mathbf{b}_j^g - \mathbf{b}_i^g\|^2_{\boldsymbol{\Sigma}^{bgd}} + \|\mathbf{b}_j^a - \mathbf{b}_i^a\|^2_{\boldsymbol{\Sigma}^{bad}} \tag{11.34}
$$

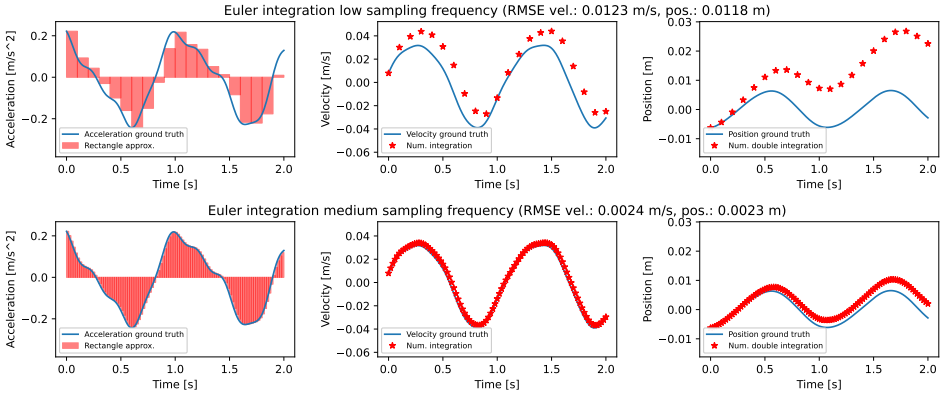

Figure 11.2 Example of Euler integration (with known initial conditions) using low (top row) and high (bottom row) sampling frequencies.

11.2.3 Advanced Preintegration Techniques

In this section, we look at some limitations of the standard preintegration approach and explore newer alternatives. We first look at the underlying signal and motion assumptions embedded in (11.14). Then we go through various works that alleviate these assumptions leading to more accurate preintegrated measurements, thus improved localization and mapping accuracy when used for aided inertial navigation. Note that we do not detail the derivation of each of the methods and invite the reader to refer to the corresponding papers for a more extensive treatment.

11.2.3.1 Numerical Integration Accuracy

As described in the previous section, standard preintegration relies on the Euler method to integrate inertial signals into rotation, velocity, and position pseudo-measurements at discrete times. This approach is fast and efficient but introduces integration error (hence drift) in the preintegration process. In short, the Euler method consists in applying the rectangle rule to a signal to numerically obtain its integral. As illustrated in Figure 11.2 (left), it means that the signal is approximated with piecewise constant chunks sampled at a given frequency. In the context of inertial systems, the samples correspond to accelerometer or gyroscope measurements.

With a fairly low sampling frequency, the piecewise constant assumption does not accurately represent the input signal. Consequently, the double integration rapidly accumulates error (Figure 11.2 (top)). A possible workaround consists in increasing the sampling frequency of the signal (Figure 11.2 (bottom)). However, in real-world inertial navigation problems, the sampling frequency is limited by the hardware characteristics of the inertial sensor.

In [638], the authors propose to use GP regression[7] as a mean to virtually upsample the input signal at any chosen timestamp for both the gyroscope and accelerometer data. While improving over standard preintegration, such an approach still performs numerical integration based on the piecewise constant assumption and does not fully leverage the continuous nature of GP models. Below, we review more sophisticated integration approaches.

11.2.3.2 Continuous Acceleration Preintegration

Another way to reduced the integration error is to leverage continuous representations —which are not limited to discrete timestamps— that better approximate the true inertial signals and perform analytical integration. Atop the gain in accuracy, continuous representations allow for asynchronous query of the preintegrated measurements. This is especially useful when performing inertial-aided state estimation with other sensors that are not hardware-synchronized or that have completely asynchronous sampling processes (*e.g.,* event cameras).

A challenging component of preintegration is dealing with the space of rotations. The non-commutative nature of rotation operations prevents the use of numerous tools available for classic Riemann integration. Accordingly, several works have dissociated the rotational and translation parts of preintegration. In this subsection, we first explore the translation component of preintegration using continuous representations while assuming solved rotation integration. Continuous integration over the rotation space will be addressed in the following subsection.

In [308], after using a zeroth-order integrator [1109] to integrate the gyroscope measurements, the authors present a continuous formulation of the velocity and position preintegrated measurements by solving the continuous-time system of differential equation (LTV) assuming constant accelerometer measurements or constant *local* acceleration (the two different models are presented in [308]). Compared with the standard preintegration [343] that consider constant global acceleration, the work [308] demonstrates that the assumption of constant local acceleration is more representative of real scenarios leading to an overall VIO accuracy improvement of around 5% on the EuRoc dataset [137] over both the standard preintegration and the constant accelerometer measurement model.

In order to loosen the constant acceleration motion model assumptions, one can approximate the input data with analytically integrable functions. Assuming that the rotational part of the preintegration is solved, the authors in [639] represent the rotation-corrected accelerometer measurements $\hat{\mathbf{v}}_k$, defined as $\hat{\mathbf{v}}_k = \Delta\boldsymbol{R}_{ik}\tilde{\mathbf{v}}_k$, in a continuous manner. We show in Figure 11.3 the accuracy gain of integration with both piecewise-linear and GP-based continuous representations compared to the Euler method shown in Figure 11.2. With the piecewise-linear approximation, the first integral (from $\hat{\mathbf{v}}_k$ to $\Delta\mathbf{v}_{ik}$) corresponds to the classic trapezoidal rule

[7] GP regression is a non-parametric, probabilistic approach for interpolation. We invite the reader to refer to [916] for a deeper understanding of GP regression.

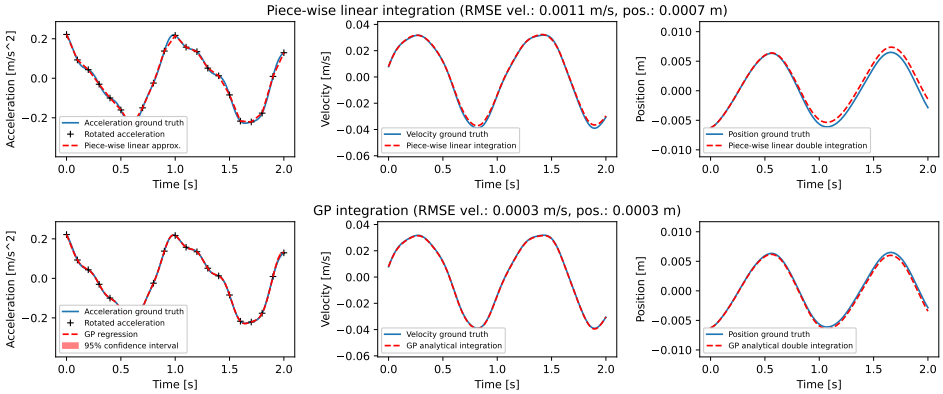

Figure 11.3 Top row: Continuous integration with piecewise-linear approximation (corresponding to constant-jerk motion assumption). Bottom row: Model-free integration with Gaussian Process regression.

for numerical integration. This model can be interpreted as a constant-jerk motion model and already provides a significant accuracy gain compared to the Euler method. Going further in the quest for model-less integration, using GP regression with $\hat{\dot{\mathbf{v}}} \sim \mathcal{GP}\left(0, k_{\dot{\mathbf{v}}}(t,t')\mathbf{I}\right)$ and $k_{\dot{\mathbf{v}}}(t,t')$ is the square exponential covariance kernel function, the direct analytical inference of the integral (and double integral) of $\hat{\dot{\mathbf{v}}}$ is enabled by the application of linear operators on GPs [974]. Accordingly, the method does not rely on any explicit motion model as the square exponential kernel is infinitely differentiable. The bottom row of Figure 11.3 shows how the non-parametric GP model improves the integration accuracy compared to the piecewise-linear method. Note that the kernel's hyperparameters control the smoothness of the signal and can be learned from the data or be set with an educated guess.

11.2.3.3 Continuous Rotation Preintegration

Looking at the accuracy gain brought by continuous representations for the translation and velocity preintegration, we naturally want to extend the concept to the rotation part. However, integrating over the space of rotations is challenging as the rotations $\boldsymbol{R}$ belong to the $SO(3)$ Lie group, which is not an Euclidean space. Properties like the commutativity of the group operation do not hold for rotations. Indeed, the product integral

$$\boldsymbol{R}_{\mathrm{B}}^{\mathrm{W}}(t+\Delta t) = \boldsymbol{R}_{\mathrm{B}}^{\mathrm{W}}(t) \prod_{t}^{t+\Delta t} \mathrm{Exp}\left(\boldsymbol{\omega}_{\mathrm{WB}}^{\mathrm{B}}(\tau)\right)^{d\tau} \tag{11.35}$$

that solves the kinematic model (11.8) does not have a known generic solution [115] and novel approaches are required to perform continuous model-less integration over the space of rotations.

In response to this challenge, the authors of [637] propose to leverage the rotation vector representation $\mathbf{r}(t)$ in the Lie algebra (with $\boldsymbol{R}(t) = \mathrm{Exp}(\mathbf{r}(t))$) as a linear vector space to perform continuous integration using linear tools. In that space, the system's dynamics is

$$\dot{\mathbf{r}} = \left(\mathtt{J}_r(\mathbf{r})\right)^{-1} \boldsymbol{\omega}_{\mathrm{WB}}^{\mathrm{B}}, \tag{11.36}$$

where $\mathtt{J}_r(\mathbf{r})$ is the right-hand Jacobian of SO(3) evaluated at $\mathbf{r}$. Unfortunately, neither $\mathbf{r}$ nor $\dot{\mathbf{r}}$ are directly observed by the IMU. The key idea of [637] is to model $\dot{\mathbf{r}}$ with a GP and a set of virtual observations $\dot{\mathbf{r}}_{t_\bullet}$ to represent the continuous rotation vector function $\mathbf{r}$ via the use of linear operators on GPs. Intuitively, the virtual observations can be interpreted as control points of the continuous rotational dynamics. These are estimated through a non-linear least-square optimisation problem with residuals based on (11.36) and the gyroscope measurements as observations of $\boldsymbol{\omega}_{\mathrm{WB}}^{\mathrm{B}}$. This results in a model-less approach to continuous rotation preintegration and yields accuracy improvements of at least one order of magnitude over the standard discrete preintegration.

This continuous approach shares a lot of similarities with the STEAM continuous-time state estimation detailed in [54] as both operate in the Lie algebra to perform GP-based interpolation. A major difference is the use of the square exponential kernel that results in a dense linear system, compared to the sparse Markovian approach used in STEAM. However, for the sake of IMU preintegration the length of an integration window is generally short enough that solving a dense system is not an issue. The concept of optimized inducing values is extended in [384] to also estimate the rotation-corrected acceleration along with the rotation vector. This allows to correlate the rotation and translation parts of the preintegrated measurement covariance matrix.

11.3 Observability of Aided Inertial Navigation

As we mentioned earlier in this chapter, due to measurement noise, biases, and inaccuracies of numerical integration, pure inertial odometry may drift quickly, in particular when using low-fidelity inertial sensors. A common approach to reduce the drift is to pair the IMU with exteroceptive sensors, *e.g.,* cameras or LiDARs, leading to *aided INS (AINS).* In many cases, the introduction of exteroceptive sensors further increases the size of the state we have to estimate, *e.g.,* by adding extra variables corresponding to external landmarks, hence a natural question to ask is whether the sensor data is *sufficient* to unambiguously estimate the SLAM state of the system. This is the goal of the *observability analysis*, which ascertains whether the information provided by the available measurements is sufficient for estimating the state/parameters without ambiguity [123, 461].

The observability analysis is typically performed by deriving linearized measurement models and computing the *observability matrix*, which is closely related to the

Fisher information (and covariance) matrix of the state estimate [495, 493]. When a system is observable, the observability matrix is full-rank; if not, as this matrix describes the information available in the measurements, studying its nullspace enables us to gain insights about the directions in the state space along which the estimator lacks sufficient information. The results of the observability analysis can be used to improve estimation consistency [1229, 464, 660], determine the minimal measurements needed to initialize an estimator [464, 744], and also identify degenerate motions that cause additional unobservable directions and should be avoided or alerted if possible in practice [1230]. For these reasons, significant research efforts have been devoted to the observability analysis of AINS [1229], and in particular visual-inertial systems (*e.g.,* [465, 661, 1230]).

In this section we discuss observability properties when the sensors used to aid the IMU produces geometric features, including points, lines, and planes. This general treatment allows discussing observability for a broad range of sensors, including cameras and LiDARs, and understanding degenerate configurations. In particular, Section 11.3.1 introduces linearized models assuming exteroceptive measurements of geometric landmarks, Section 11.3.2 uses these models to perform the observability analysis, and Section 11.3.3 discusses degenerate configurations.

11.3.1 Linearized Measurement Models

We describe the measurement models assuming that the sensor (*e.g.,* camera, LiDAR) used to aid the IMU produces geometric features; in other words, we focus on SLAM and odometry front-ends which produce landmark-based representations. While most AINS use point features, in particular when relying on cameras (e.g., [464, 660, 652, 903, 383, 343]), line and plane features can be utilized when available (e.g., [607, 463, 421, 1231]). In such a case, we may need to augment the to-be-estimated state vector with all these different geometric features. Specifically, the AINS state that we are trying to estimate (at each time step) includes both the state of the robot $\boldsymbol{x}_{\text{B}}$ and the state of external features $\boldsymbol{x}_{\text{f}}^{\text{W}}$ (expressed in the world frame):

$$\boldsymbol{x} = \{\boldsymbol{R}_{\text{B}}^{\text{W}}, \mathbf{b}^{g}, \mathbf{v}^{\text{W}}, \mathbf{b}^{a}, \mathbf{p}^{\text{W}}, \boldsymbol{x}_{\text{f}}^{\text{W}}\} \tag{11.37}$$

where $\boldsymbol{R}_{\text{B}}^{\text{W}}$ is the rotation of the body frame B with respect to the world frame W, and $\mathbf{p}^{\text{W}}, \mathbf{v}^{\text{W}}$ are the robot position and velocity expressed in the world frame, respectively, while $\mathbf{b}^{g}, \mathbf{b}^{a}$ are the gyroscope and accelerometer biases in the body frame. The features $\boldsymbol{x}_{\text{f}}^{\text{W}}$ can be either points, lines, or planes (or a combination thereof) and are expressed in the world frame.

For the ensuing observability analysis, we need both the system dynamic model (which is related to the accelerations and angular rate measurements of the IMU) and the exteroceptive measurement model. Below, we start by reviewing the INS

kinematic model —building on the IMU equations introduced in the previous section— and then consider exteroceptive measurement equations.

11.3.1.1 Linearized IMU Kinematic Model

The IMU-based kinematic model is given by (*cf.* (11.8) and (11.32)):

$$\dot{\boldsymbol{R}}_{\mathrm{B}}^{\mathrm{W}} = \boldsymbol{R}_{\mathrm{B}}^{\mathrm{W}}(\boldsymbol{\omega}_{\mathrm{WB}}^{\mathrm{B}})^{\wedge}, \quad \dot{\mathbf{v}}^{\mathrm{W}} = \dot{\mathbf{v}}^{\mathrm{W}}, \quad \dot{\mathbf{p}}^{\mathrm{W}} = \mathbf{v}^{\mathrm{W}}, \tag{11.38}$$

$$\dot{\mathbf{b}}^{g}(t) = \boldsymbol{\eta}^{bg}, \quad \dot{\mathbf{b}}^{a}(t) = \boldsymbol{\eta}^{ba} \tag{11.39}$$

where $\boldsymbol{\eta}^{bg}$ and $\boldsymbol{\eta}^{ba}$ are the zero-mean Gaussian noises driving the gyroscope and accelerometer biases (which are modeled as random walks). In order to perform the observability analysis, we linearize the above nonlinear system and obtain the following continuous-time linearized error-state dynamical system:

$$\dot{\tilde{\boldsymbol{x}}}(t) \simeq \begin{bmatrix} \mathbf{F}_c(t) & \mathbf{0}_{15\times n_{\mathrm{f}}} \\ \mathbf{0}_{n_{\mathrm{f}}\times 15} & \mathbf{0}_{n_{\mathrm{f}}} \end{bmatrix} \tilde{\boldsymbol{x}}(t) + \begin{bmatrix} \mathbf{G}_c(t) \\ \mathbf{0}_{n_{\mathrm{f}}\times 12} \end{bmatrix} \boldsymbol{\eta}(t) =: \mathbf{F}(t)\tilde{\boldsymbol{x}}(t) + \mathbf{G}(t)\boldsymbol{\eta}(t) \tag{11.40}$$

where the error-state vector $\tilde{\boldsymbol{x}} = \{\tilde{\boldsymbol{\theta}}, \tilde{\mathbf{b}}^{g}, \tilde{\mathbf{v}}^{\mathrm{W}}, \tilde{\mathbf{b}}^{a}, \tilde{\mathbf{p}}^{\mathrm{W}}, \tilde{\boldsymbol{x}}_{\mathrm{f}}^{\mathrm{W}}\}$ (expressed as a column vector) represents the deviation from the linearization point (*e.g.,* $\tilde{\mathbf{b}}^{g}$ is the change of the bias with respect to the linearization point), and for the rotation component we use the tangent-space representation $\tilde{\boldsymbol{\theta}}$ at the linearization point.[8] In (11.40), n_{f} is the dimension of $\tilde{\boldsymbol{x}}_{\mathrm{f}}^{\mathrm{W}}$, $\mathbf{F}_c(t)$ and $\mathbf{G}_c(t)$ are the continuous-time linearized transition matrix and the noise Jacobian matrix for the IMU state, respectively, and $\boldsymbol{\eta}(t)$ is the stacked noise, including both $\boldsymbol{\eta}^{bg}$ and $\boldsymbol{\eta}^{ba}$ as well as the IMU noise which arises when substituting the actual acceleration and rotation rates in (11.38) with the accelerometer and gyroscope measurements (*cf.* with derivation in Section 11.2.1).

As in practice AINS estimators are typically implemented in discrete time, the discrete-time dynamic model is needed and can be derived by computing its state transition matrix $\boldsymbol{\Phi}_{(k+1,k)}$ from time t_k to t_{k+1}, based on $\dot{\boldsymbol{\Phi}}_{(k+1,k)} = \mathbf{F}(t_k)\boldsymbol{\Phi}_{(k+1,k)}$ with identity as the initial condition:

$$\boldsymbol{\Phi}_{(k+1,k)} = \begin{bmatrix} \boldsymbol{\Phi}_{11} & \boldsymbol{\Phi}_{12} & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_{n_{\mathrm{f}}\times 3} \\ \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_{n_{\mathrm{f}}\times 3} \\ \boldsymbol{\Phi}_{31} & \boldsymbol{\Phi}_{32} & \mathbf{I}_3 & \boldsymbol{\Phi}_{34} & \mathbf{0}_3 & \mathbf{0}_{n_{\mathrm{f}}\times 3} \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_{n_{\mathrm{f}}\times 3} \\ \boldsymbol{\Phi}_{51} & \boldsymbol{\Phi}_{52} & \boldsymbol{\Phi}_{53} & \boldsymbol{\Phi}_{54} & \mathbf{I}_3 & \mathbf{0}_{n_{\mathrm{f}}\times 3} \\ \mathbf{0}_{3\times n_{\mathrm{f}}} & \mathbf{0}_{3\times n_{\mathrm{f}}} & \mathbf{0}_{3\times n_{\mathrm{f}}} & \mathbf{0}_{3\times n_{\mathrm{f}}} & \mathbf{0}_{3\times n_{\mathrm{f}}} & \mathbf{I}_{n_{\mathrm{f}}} \end{bmatrix} \tag{11.41}$$

where the (i, j) block $\boldsymbol{\Phi}_{ij}$ can be found analytically or numerically [464].

[8] Intuitively, to linearize the rotation variables, we use the fact that we can rewrite any rotation $\boldsymbol{R}_{\mathrm{B}}^{\mathrm{W}}$ as a perturbation of the rotation at the linearization point $\hat{\boldsymbol{R}}_{\mathrm{B}}^{\mathrm{W}}$: $\boldsymbol{R}_{\mathrm{B}}^{\mathrm{W}} = \hat{\boldsymbol{R}}_{\mathrm{B}}^{\mathrm{W}}\boldsymbol{R}_{\mathrm{B}}^{\mathrm{W}}(\tilde{\boldsymbol{\theta}})$, where $\tilde{\boldsymbol{\theta}}$ is a suitable tangent-space vector. Then we can use the following small-angle approximation to linearize the expression: $\boldsymbol{R}_{\mathrm{B}}^{\mathrm{W}} = \hat{\boldsymbol{R}}_{\mathrm{B}}^{\mathrm{W}}\boldsymbol{R}_{\mathrm{B}}^{\mathrm{W}}(\tilde{\boldsymbol{\theta}}) \simeq \hat{\boldsymbol{R}}_{\mathrm{B}}^{\mathrm{W}}(\mathbf{I} + \tilde{\boldsymbol{\theta}}^{\wedge})$.

11.3.1.2 Exteroceptive Measurement Models

Now we present the measurement models of different geometric features and their linearized models, which are essential for the linearized AINS observability analysis.

Point Features. Consider point feature measurements provided by exteroceptive sensors (such as monocular/stereo camera, acoustic sonar, and LiDAR). These can generally be modeled as range and/or bearing observations, which are functions of the relative position of the feature in the sensor frame C:

$$\boldsymbol{z}_p = \underbrace{\begin{bmatrix} \lambda_r & \mathbf{0}_{1\times 2} \\ \mathbf{0}_{2\times 1} & \lambda_b \mathbf{I}_2 \end{bmatrix}}_{\boldsymbol{\Lambda}} \begin{bmatrix} z_r \\ \boldsymbol{z}_b \end{bmatrix} = \boldsymbol{\Lambda} \begin{bmatrix} \|\mathbf{p}_{\mathrm{f}}^{\mathrm{C}}\| + \eta^r \\ h_b\left(\mathbf{p}_{\mathrm{f}}^{\mathrm{C}}\right) + \boldsymbol{\eta}^b \end{bmatrix} \tag{11.42}$$

where $\mathbf{p}_{\mathrm{f}}^{\mathrm{C}} = \boldsymbol{R}_{\mathrm{W}}^{\mathrm{C}}\left(\mathbf{p}_{\mathrm{f}}^{\mathrm{W}} - \mathbf{p}_{\mathrm{C}}^{\mathrm{W}}\right)$ is the position of the feature in the sensor frame, and z_r and $\boldsymbol{z}_b$ denote range and bearing measurements, respectively. In particular, $h_b(\cdot)$ is a generic bearing measurement function whose actual form depends on the particular sensor used. $\boldsymbol{\Lambda}$ is a measurement selection matrix, with binary entries λ_r and λ_b; for example, if $\lambda_b = 1$ and $\lambda_r = 1$, then $\boldsymbol{z}_p$ contains both range and bearing measurements. η^r and $\boldsymbol{\eta}^b$ are the measurement noises and are assumed to be additive for simplicity. Linearizing (11.42) with the chain rule of differentiation at the current state estimate yields the following measurement error equation:

$$\tilde{\boldsymbol{z}}_p = \boldsymbol{z}_p - \hat{\boldsymbol{z}}_p \simeq \boldsymbol{\Lambda} \begin{bmatrix} \frac{\partial z_r}{\partial \mathbf{p}_{\mathrm{f}}^{\mathrm{C}}} \frac{\partial \mathbf{p}_{\mathrm{f}}^{\mathrm{C}}}{\partial \boldsymbol{x}} \Big|_{\hat{\boldsymbol{x}}} \tilde{\boldsymbol{x}} + \eta^r \\ \frac{\partial \boldsymbol{z}_b}{\partial \mathbf{p}_{\mathrm{f}}^{\mathrm{C}}} \frac{\partial \mathbf{p}_{\mathrm{f}}^{\mathrm{C}}}{\partial \boldsymbol{x}} \Big|_{\hat{\boldsymbol{x}}} \tilde{\boldsymbol{x}} + \boldsymbol{\eta}^b \end{bmatrix} =: \boldsymbol{\Lambda} \begin{bmatrix} \mathbf{H}_r \\ \mathbf{H}_b \end{bmatrix} \mathbf{H}_{\mathrm{f}} \tilde{\boldsymbol{x}} + \boldsymbol{\Lambda} \begin{bmatrix} \eta^r \\ \boldsymbol{\eta}^b \end{bmatrix} =: \mathbf{H}_x \tilde{\boldsymbol{x}} + \boldsymbol{\eta}^p \tag{11.43}$$

where $\hat{\boldsymbol{z}}_p$ is the measurement at the linearization point. Depending on the selection matrix $\boldsymbol{\Lambda}$, the Jacobian $\mathbf{H}_x$ may include the range-only measurement Jacobian $\mathbf{H}_r$ ($\lambda_r = 1$, $\lambda_b = 0$), the bearing-only Jacobian $\mathbf{H}_b$ ($\lambda_r = 0$, $\lambda_b = 1$), or both.

Line Features. Given two 3D points $\mathbf{p}_1^W$ and $\mathbf{p}_2^W$, we can represent the line passing through the two points using its Plücker coordinates:

$$\mathbf{l}^{\mathrm{W}} = \begin{bmatrix} \mathbf{n}_\ell^W \\ \mathbf{v}_\ell^W \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1^W \times \mathbf{p}_2^W \\ \mathbf{p}_2^W - \mathbf{p}_1^W \end{bmatrix} \tag{11.44}$$

where $\mathbf{n}_\ell^W$ is the line moment that encodes the normal direction of the plane defined by the two points and the origin, and $\mathbf{v}_\ell^W$ is the line direction vector which can be normalized to a unit vector if needed. Note that the distance from the origin to the line can be computed as $d_\ell^{\mathrm{W}} = \frac{\|\mathbf{n}_\ell^W\|}{\|\mathbf{v}_\ell^W\|}$, and the above Plücker coordinate —expressed in the world frame— can be transformed to the camera frame as [1027]:

$$\begin{bmatrix} \mathbf{n}_\ell^{\mathrm{C}} \\ \mathbf{v}_\ell^{\mathrm{C}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{C}}^{\mathrm{W}\top} & -\boldsymbol{R}_{\mathrm{C}}^{\mathrm{W}\top}(\mathbf{p}_{\mathrm{C}}^{\mathrm{W}})^\wedge \\ \mathbf{0} & \boldsymbol{R}_{\mathrm{C}}^{\mathrm{W}\top} \end{bmatrix} \begin{bmatrix} \mathbf{n}_\ell^{\mathrm{W}} \\ \mathbf{v}_\ell^{\mathrm{W}} \end{bmatrix} \tag{11.45}$$

We now consider a case where the 3D line is observed in 2D images. Specifically, given two endpoints of a line segment in the image: $\mathbf{q}_1 := [u_1, v_1, 1]^\mathsf{T}$ and $\mathbf{q}_2 :=$

$[u_2, v_2, 1]^\mathsf{T}$, we derive the 2D visual line measurement model as the distances of these two endpoints to the back-projected 3D Plücker line onto the image plane [1310]. To this end, we transform the 3D line from the world frame to the current camera frame via (11.45) and then project it onto the image with the known intrinsic parameters of the camera [1027]:

$$\boldsymbol{\ell} = \underbrace{\begin{bmatrix} f_2 & 0 & 0 \\ 0 & f_1 & 0 \\ -f_2 c_1 & -f_1 c_2 & f_1 f_2 \end{bmatrix}}_{\mathbf{K}} \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 \end{bmatrix} \begin{bmatrix} \mathbf{n}_\ell^C \\ \mathbf{v}_\ell^C \end{bmatrix} =: \begin{bmatrix} \ell_1 \\ \ell_2 \\ \ell_3 \end{bmatrix} \tag{11.46}$$

where $\mathbf{K}$ is the canonical projection Plücker matrix and f_1, f_2, c_1 and c_2 are the standard camera intrinsic parameters. Note that only the moment vector $\mathbf{n}_\ell^C$ in the Plücker coordinates is involved in the above projection, which implies that the line range and orientation contained in $\mathbf{v}_\ell^C$ are not measurable. Therefore, the distances of the two endpoints of the line segment to the projected line $\boldsymbol{\ell}$ in the image can be finally computed and used as the line feature measurements:

$$\boldsymbol{z}_\ell = \begin{bmatrix} \frac{\mathbf{q}_1^\top \boldsymbol{\ell}}{\sqrt{\ell_1^2 + \ell_2^2}} \\ \frac{\mathbf{q}_2^\top \boldsymbol{\ell}}{\sqrt{\ell_1^2 + \ell_2^2}} \end{bmatrix} + \boldsymbol{\eta}^\ell \tag{11.47}$$

where $\boldsymbol{\eta}^\ell$ is the measurement noise. Similarly, with the chain rule of differentiation, we can linearize (11.47) with respect to the state and obtain the measurement Jacobian: $\mathbf{H}_x = \frac{\partial \boldsymbol{z}_\ell}{\partial \boldsymbol{\ell}} \frac{\partial \boldsymbol{\ell}}{\partial \boldsymbol{x}}$.

Plane Features. A 3D plane can be parameterized by its distance to the origin and normal direction in the world frame: $\boldsymbol{\pi}^\mathrm{W} = \begin{bmatrix} \mathbf{n}_\pi^\mathrm{W} \\ d_\pi^\mathrm{W} \end{bmatrix}$, which can be transformed to the local sensor frame where the plane feature is typically detected:

$$\begin{bmatrix} \boldsymbol{n}_\pi^\mathrm{C} \\ d_\pi^\mathrm{C} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_\mathrm{W}^C & \mathbf{0}_{3\times 1} \\ -(\mathbf{p}_C^\mathrm{W})^\mathsf{T} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{n}_\pi^\mathrm{W} \\ d_\pi^\mathrm{W} \end{bmatrix} \tag{11.48}$$

Without loss of generality, we consider a plane feature $(\boldsymbol{n}_\pi^\mathrm{C}, d_\pi^\mathrm{C})$ is extracted from point clouds (*e.g.,* LiDAR or depth sensors), and employ the closes point $\mathbf{p}_\pi^\mathrm{C} = d_\pi^\mathrm{C} \boldsymbol{n}_\pi^\mathrm{C}$ from the plane to the origin as the plane representation in the AINS state vector [382].

$$\boldsymbol{z}_\pi = d_\pi^\mathrm{C} \boldsymbol{n}_\pi^\mathrm{C} + \boldsymbol{\eta}^\pi = \mathbf{p}_\pi^\mathrm{C} + \boldsymbol{\eta}^\pi \tag{11.49}$$

where $\boldsymbol{\eta}^\pi$ is the plane measurement noise. Linearization of (11.49) yields the plane measurement Jacobian $\mathbf{H}_x = \frac{\partial \boldsymbol{z}_\pi}{\partial \mathbf{p}_\pi^\mathrm{C}} \frac{\partial \mathbf{p}_\pi^\mathrm{C}}{\partial \boldsymbol{x}}$.

11.3.2 Observability Analysis

Based on the linearized system and measurement models presented in the previous sections, we can now perform the observability analysis. The analysis relies on the following *observability matrix* $\mathbf{M}(\hat{\boldsymbol{x}})$ to gain insights about the system (*cf.* [494]):

$$\mathbf{M}(\hat{\boldsymbol{x}}) = \begin{bmatrix} \mathbf{H}_{x_1}\boldsymbol{\Phi}_{(1,1)} \\ \mathbf{H}_{x_2}\boldsymbol{\Phi}_{(2,1)} \\ \vdots \\ \mathbf{H}_{x_k}\boldsymbol{\Phi}_{(k,1)} \end{bmatrix} \tag{11.50}$$

where $\mathbf{H}_{x_k}$ stacks the Jacobians for all the measurements (of points, lines, or planes) collected at discrete time k, and the notation $\mathbf{M}(\hat{\boldsymbol{x}})$ stresses the fact that the observability matrix depends on the linearization point $\hat{\boldsymbol{x}}$. The nullspace $\mathcal{U}$ of this matrix, *i.e.,* the span of the null vectors $\text{span}([\cdots \ \boldsymbol{u}_i \ \cdots]) = \mathcal{U}$ such that $\mathbf{M}(\boldsymbol{x})\boldsymbol{u}_i = \mathbf{0}$, describes the unobservable subspace of AINS. If the nullspace is empty, the system is fully observable. It has been shown in [1229] that AINS in general has 4 unobservable directions (*i.e.,* it has four independent vectors in the null space $\mathcal{U}$), describing the fact that the global 3D position and global yaw are unobservable from IMU measurements and local observations of previously unknown landmarks.

To understand the structure of the 4-dimensional null space, we consider the case where all three types of geometric features (*i.e.,* a single point, line, and plane) are in the state vector: $\boldsymbol{x}_{\text{f}}^{\text{W}} = \{\mathbf{p}_{\text{f}}^{\text{W}}, \mathbf{l}^{\text{W}}, \boldsymbol{\pi}^{\text{W}}\}$, and the exteroceptive measurements include: $\boldsymbol{z} = \{\boldsymbol{z}_p, \boldsymbol{z}_\ell, \boldsymbol{z}_\pi\}$ (*cf.* (11.42), (11.47), and (11.49)). By computing the related system and measurement Jacobians (*i.e.,* $\mathbf{H}_{x_i}$ and $\boldsymbol{\Phi}_{(i,1)}$) and substituting them into (11.50), we can build the corresponding linearized AINS observability matrix $\mathbf{M}$. By mathematically computing the nullspace of this matrix $\text{null}(\mathbf{M})$, we should be able to find the following four null-vectors (*cf.* [1229]):

$$\text{null}(\mathbf{M}) = \text{span}[\boldsymbol{u}_1 \ \boldsymbol{u}_{2:4}] = \text{span}\begin{bmatrix} \boldsymbol{u}_g & \mathbf{0}_{12\times 3} \\ -\mathbf{p}_1^{\text{W}} \times \mathbf{g}^{\text{W}} & \mathbf{I}_3 \\ -\mathbf{p}_{\text{f}}^{\text{W}} \times \mathbf{g}^{\text{W}} & \mathbf{I}_3 \\ -\mathbf{g}^{\text{W}} & \frac{\mathbf{v}_\ell^{\text{W}}}{d_\ell^{\text{W}}||\mathbf{v}_\ell^{\text{W}}||}(\boldsymbol{R}_\ell^{\text{W}}\mathbf{e}_1)^\mathsf{T} \\ 0 & -(\boldsymbol{R}_\ell^{\text{W}}\mathbf{e}_3)^\mathsf{T} \\ -d_\pi^{\text{W}}\mathbf{n}_\pi^{\text{W}} \times \mathbf{g}^{\text{W}} & \mathbf{n}_\pi^{\text{W}}(\boldsymbol{R}_\pi^{\text{W}}\mathbf{e}_3)^\mathsf{T} \end{bmatrix} \tag{11.51}$$

where $\boldsymbol{u}_g = \begin{bmatrix} (\boldsymbol{R}_{\text{W}}^{\text{C}_1}\mathbf{g}^{\text{W}})^\mathsf{T} & \mathbf{0}_{1\times 3} & -(\mathbf{v}_1^{\text{W}} \times \mathbf{g}^{\text{W}})^\mathsf{T} & \mathbf{0}_{1\times 3} \end{bmatrix}^\mathsf{T}$, $\mathbf{p}_1^{\text{W}}$ refers to the sensor position at the time $k = 1$, $\boldsymbol{R}_{\text{W}}^{\text{C}_1}$ is the rotation matrix from the sensor frame C_1 at time $k = 1$ to the world frame W, while $\boldsymbol{R}_\pi^{\text{W}}$ is a rotation matrix built using the plane normal vector $\mathbf{n}_\pi^{\text{W}}$ using Gram–Schmidt orthonormalization (*cf.* (11.6)), and $\boldsymbol{R}_\ell^{\text{W}} = \begin{bmatrix} \frac{\mathbf{n}_\ell^{\text{W}}}{||\mathbf{n}_\ell^{\text{W}}||} & \frac{\mathbf{v}_\ell^{\text{W}}}{||\mathbf{v}_\ell^{\text{W}}||} & \frac{\mathbf{n}_\ell^{\text{W}}}{||\mathbf{n}_\ell^{\text{W}}||} \times \frac{\mathbf{v}_\ell^{\text{W}}}{||\mathbf{v}_\ell^{\text{W}}||} \end{bmatrix}$ is the rotation matrix constructed with the line normal and line direction. It is possible to see that the first null vector $\boldsymbol{u}_1$

is related to the rotation around the gravity (and hence the yaw) and $\boldsymbol{u}_{2:4}$ to the motion of the robot. Readers are referred to [1229, 1228] for more detailed analysis.

In summary, the fact that the observability matrix admits a 4-dimensional null space (*cf.* (11.51)) correctly describes the fact that the global position and yaw of the system are not observable. Intuitively, none of the measurements (IMU data, measurements of unknown point, line, or plane landmarks) convey information about the global frame, with the exception of roll and pitch, which are observable from the accelerometer measurements of the gravity direction. This unobservability is common in SLAM[9] and it not pathological: it only means that we can arbitrarily set the yaw and 3D origin of our world frame since we only have relative measurements for those variables. This unobservability would disappear when adding a sensor providing absolute measurements, *e.g.,* a GPS. More concerning is that fact that for certain motions (and linearization points), the null space of the observability matrix can grow larger, creating additional unobservable dimensions. We explore this phenomenon below.

11.3.3 Degenerate Motions

Certain types of motions might induce additional unobservable directions for AINS (*i.e.,* in addition to the 4 expected ones that we discussed above). This is of practical importance since these degenerate motions might lead to large errors in some directions of the state space and lead to navigation failures. The degenerate motions of AINS are summarized in Table 11.1 (see [1229] for a full derivation). Specifically, pure translation is degenerate for all feature types, causing the full global rotation to become unobservable. Intuitively, if the system is not rotating, we might confuse gravity measurements with accelerometer biases, hence making the roll and pitch no longer observable. The other three degenerate motions, namely constant acceleration (including the case of constant velocity, where the acceleration is set to zero), pure rotation, and motion in the direction of the feature (for the case where we have a single point feature), cause the scale to be unobservable for the case of monocular camera (*i.e.,* bearing-only measurements). However, constant acceleration causes the whole system (*i.e.,* position, velocity, acceleration bias, and features) scale to be unobservable, while pure rotation and moving toward a feature only make the feature scale unobservable. Note that these three degenerate motions hold only if the distance from the sensor to the feature is significantly larger than the extrinsic translation between the sensor and robot body (if they do not coincide), *i.e.,* $||\mathbf{p}_{\mathrm{f}}^{\mathrm{C}}|| >> ||\mathbf{p}_{\mathrm{B}}^{\mathrm{C}}||$, which typically is the case in practice.

[9] Without using an IMU, the null space for a landmark-based SLAM problem would be at least 6-dimensional, capturing the fact that without an IMU, the entire 3D rotation (in addition to the 3D position) of the system is unobservable.

Table 11.1 *Degenerate motions of AINS.*

Motion	*Sensor*	*Unobservable*
1. Pure translation	General	Global orientation
2. Constant acceleration	Mono cam	System scale
3. Pure rotation	Mono cam	Feature scale
4. Moving toward point feature	Mono cam	Feature scale

11.4 Visual-Inertial Odometry and Practical Considerations

As mentioned above, inertial measurements are typically fused with data from other sensors to mitigate the odometry drift. In this section, we particularly focus on the case where visual measurements from a camera are fused with IMU measurements using factor graphs.[10] Camera and IMU are a popular combination, since the are both inexpensive, lightweight, and low-power sensors. Moreover, they are complementary sensors, where the IMU is able to capture quick acceleration and rotations, while cameras are able to provide rich observations of the surrounding environment. On one side, the use of cameras largely reduces the drift as compared to pure inertial odometry; on the other side, an IMU might allow observing quantities that would not be possible to estimate otherwise. In particular, when using a monocular camera for SLAM, one cannot estimate the scale of the scene without relying on prior information (in other words, the scale is unobservable), while adding an IMU allows retrieving the scale as well, as long as the motion of the robot is non-degenerate (Section 11.3.3). As we mentioned earlier in this chapter, systems comprising one or more cameras as well as an IMU are typically referred to as visual-inertial odometry (VIO) systems, and become visual-inertial SLAM systems when loop closures are incorporated.

11.4.1 Visual-Inertial Odometry

VIO systems are commonly used as a source of odometry and are often used to close control loops over trajectory tracking and control. In other applications, such as virtual reality, VIO systems are used to compensate for the motion of the user in the virtual environment. In both cases, VIO is required to produce estimates with very low-latency, typically in the order of 10-50ms. For instance, the refresh rate of the Meta Quest 3 is between 72Hz and 120Hz [7], and the VIO latency directly impacts the quality of the VR experience and is key to mitigating motion sickness. Similarly, for trajectory tracking it is important to keep the latency low since large delays might induce instability and divergence of the tracking controller.

[10] In robotics, it is also common to use inertial data in combination with other sensors, including LiDAR and radars. We postpone the discussion about these other types of inertial odometry systems to Chapters 8 and 9.

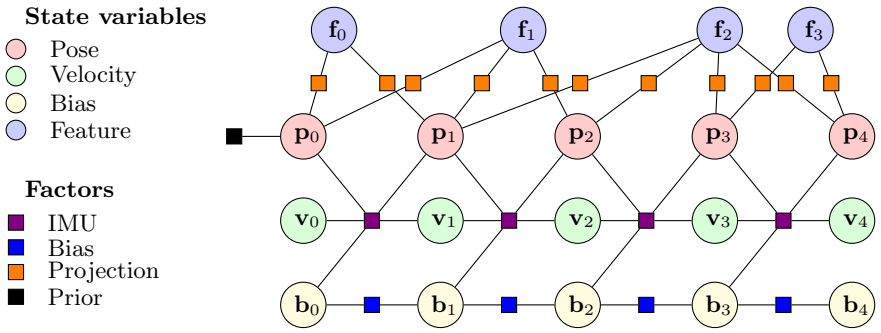

Figure 11.4 Example of factor graph used for visual-inertial odometry with preintegrated IMU factors [343]. The factor graph shows preintegrated IMU factors in violet (constraining consecutive poses, velocity, and bias), bias factors in blue (constraining the evolution of the IMU biases over time), vision factors in orange (relating camera poses and positions of external landmarks), and priors in black.

Based on these consideration, factor-graph based VIO systems typically implement a fixed-lag smoother (also called sliding-window optimization), where one only attempts to estimate states in a receding horizon (*e.g.,* the last 5-10 seconds). An example of the resulting factor graph is shown in Fig. 11.4, which shows preintegrated IMU factors in violet, bias factors in blue, vision factors in orange, and priors in black. The horizon is chosen in a way to trade-off computation with accuracy, since the longer the horizon, the larger the state space for estimation. Then factors and variables falling out of the receding horizon are gradually marginalized as time progresses. In many cases, optimized implementations also eliminate visual landmarks from the optimization using the Schur complement to further reduce the size of the state space, see, *e.g.,* [343]. An alternative to using a fixed-lag smoother is to use an incremental solver like iSAM2 (Section 1.7), which reuses computation from previous optimizations when computing an estimate at the current time. While this approach has been shown to lead to very accurate results in practice [343], it has the drawback of not providing guarantees on the latency of the system and might lead to spikes in the runtime, which is problematic for certain applications.

VIO Systems and Performance. The last decade has seen a proliferation of visual-inertial odometry/SLAM systems, many of which have open-source implementations. Popular approaches include a visual-inertial version of ORB-SLAM [792], Direct Sparse Visual-Inertial Odometry [1120], VINS-Mono [903], OpenVINS [383], Kimera [10, 947], BASALT [1122], and DM-VIO [1048]. A good VIO system has a drift below 1% of the distance traveled (*e.g.,* it accumulates an error smaller than 1m after covering a 100m trajectory), and in some cases the drift can be as low as 0.1%.

Sliding-window optimization approaches such as VINS-Mono [903] have seen tremendous success in practice. As an example, here we show how a more recent

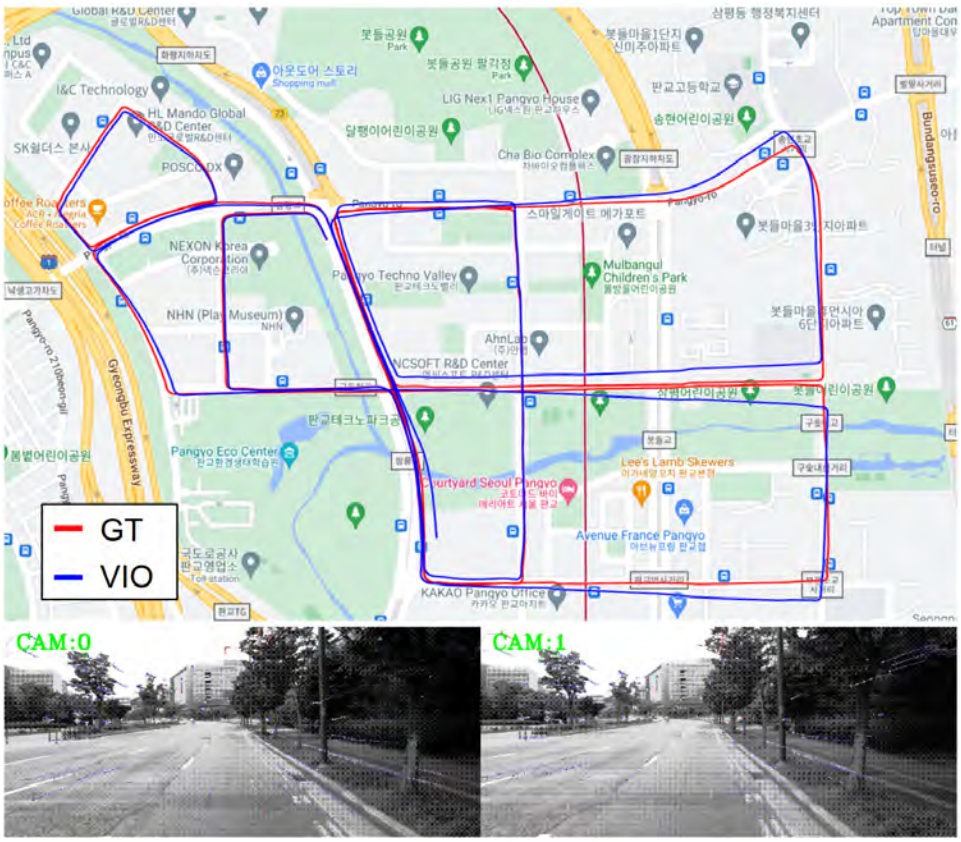

Figure 11.5 Illustration of a recent sliding-window optimization-based VIO algorithm [186] running on the KAIST urban autonomous driving dataset, sequence 38. This sequence has a total duration of 36 minutes and is 11.42 km in length. The VIO estimates and ground truth are overlaid on the Google map. Bottom are two sample images. The final ATE of the VIO (i.e., without loop closure) is 2.05 degrees and 21.2 meters (0.18%).

sliding-window-based approach, called First-Estimate Jacobian (FEJ)-based Window Bundle Adjustment (WBA)-VINS [186, 187] performs on the KAIST Urban Dataset [522]. The KAIST urban dataset focuses on autonomous driving and localization in challenging complex urban environments. The dataset was collected in Korea with a vehicle equipped with stereo camera pair, 2D/3D LiDARs, Xsens IMU, Fiber Optic Gyro (FoG), wheel encoders, and RKT GPS. The camera operates at 10Hz, while the IMU sensing rate is 100Hz. A ground-truth trajectory is also provided which is obtained from the fusion of the FoG, RKT GPS, and wheel encoders. Fig. 11.5 illustrates the FEJ-WBA-VINS [186, 187] (VIO) estimated trajectory for *sequence 38*, overlaid on a Google map alongside the ground truth (GT) trajectory. The final absolute trajectory error (ATE) for the 11.42-kilometer path

is approximately 2.05 degrees and 21.2 meters (0.18% of the trajectory traveled). Notably, these results are obtained from the pure online VIO without loop closure.

Applications of VIO to self-driving cars and other autonomous systems is discussed in [10, 11], which also discusses challenges related to feature tracking, keyframe section, and fusion of different sensor modalities (including monocular, stereo, and RGB-D camera images, as well as wheel odometry).

11.4.2 Extrinsic Calibration

In order to enable accurate aided inertial navigation, one needs to perform extrinsic calibration of the sensors. The extrinsic calibration corresponds to estimating the relative pose between the different sensors (*e.g.,* the pose of the camera with respect to the IMU). The literature provides a variety of methods to address the calibration problem. These can be mainly divided into two categories: offline and online calibration methods. Offline approaches require a calibration procedure to be performed before the system is deployed. It often involves the use of a calibration target [358], a known motion pattern [693], or a prior knowledge about the environment [638, 722]. These procedures can be more-or-less time consuming, and may require specific equipment and trained operators. Thus, while generally more accurate, offline calibration methods can be cumbersome and undesirable in some scenarios where the final system is expected to be used at large scale by non-experts. Online methods, on the other hand, do not require a specific procedure [309, 640, 1207, 1232]. Instead, the extrinsic calibration parameters are estimated as part of the state estimation problem. This offers the advantage of being able to adapt to changes in the system, such as sensor displacement, without the need for a new calibration procedure. However, online calibration methods can be less accurate than offline methods, and may render the state estimation problem more complex or even ill-posed [1230].

11.4.3 Temporal Synchronization

Another crucial aspect of inertial-aided system is the temporal synchronization of the sensor data. An erroneous synchronization that is not accounted for can lead to significant errors in the trajectory estimates and/or introduce a bias in the benchmarking metrics. The synchronization can be done either in hardware or in software. The low-level hardware approach often relies on a dedicated piece of hardware that triggers the various sensors' data acquisition based on a common clock signal via specific synchronization input pins. This is not always possible, especially when the sensors are connected to the computer via different communication protocols. Some sensors may have a built-in synchronization mechanism that can be used to synchronize the different sensors' clock without the need for a dedicated hardware

input. The use of PTP (Precision Time Protocol) is an example of such a software-based synchronization over Ethernet. Many LiDARs, radars and INS solutions can be synchronized with this protocol. Another solution is time-stamping the data at the sensor level and then aligning the timestamps in a post-processing step. This last approach is generally less accurate and robust than the aforementioned ones. If the system cannot be synchronized and post-processing is not an option (e.g., online applications), some state estimation algorithms integrate the time offset as a state variable in the estimation problem [309, 384, 1232].

11.5 New trends

While progress in inertial odometry is steadily transitioning into industry products, aided inertial navigation is still the subject of intense research.

Extended Pose Preintegration. Latest trends in inertial odometry for SLAM include the use of extended-pose manifolds and higher-order noise propagation [126] to improve the uncertainty modeling of IMU preintegration. The authors of [126] extend the preintegration theory to account for Earth's rotation with the Coriolis and centrifugal forces. The work [1130] provides an example of extended pose preintegration leveraging both a linear velocity sensor and a navigational-grade IMU. After an hour of marine navigation over a 1.8km trajectory, the authors report a translation error of around 5m.

Continuous-time State Representations. We have mostly been interested in the use of IMUs under the scope of preintegration as a mean to reduce the number of discrete state variables in our factor graphs. However, other approaches bases on continuous-time state representation can also account from many IMU measurements without increasing the dimensionality of the estimated state. We find such examples in [357] using B-splines basis functions and in [54] with GP priors. Both formulations allow to use IMU measurements at high rates in residuals based on interpolated dynamics between a fixed set of state variables. Recently, the work [136] compares the integration of IMU measurements directly as inputs in the continuous-time GP prior against using IMU measurements directly in residuals. They concluded that using inertial information as measurements of the state resulted in better odometry accuracy using a LiDAR-inertial sensor suite. Another interesting work on continuous-time representations is [666], where the authors compare the GP-based state representation from [54] with the continuous GP-based preintegration from [384] (presented earlier in this chapter). In a event-based VIO context, the authors show that the later provides a slight advantage over the former both in terms of accuracy and computational efficiency.

Proprioception-only Odometry. Recent works use proprioceptive sensors for aided inertial navigation. For odometry, works like [448] with legged robots and [819] with wheel-mounted IMUs demonstrate how some knowledge about the system's kinematics can be used to provide competitive IMU-based odometry estimates with

sub-percent positional error. In [448] the critical information is the knowledge of contact between the robot's feet and the ground, while in [819] the one-plane-rotation motion is used to constrain the IMU biases and therefore limit the dead-reckoning drift. The work [819] has been extended into a full SLAM system [1197] by detecting loop closures based on pattern recognition in the road bank angle over time, providing an interesting example of an IMU-based proprioceptive system that can perform loop closure detection and correction. It is important to note that while the use of inertial sensors generally offers better performance and robustness, dropouts or saturation of the IMU sensor can have catastrophic effects on the overall system's performance. In [274] the authors investigate the use of accelerometer data to estimate angular velocity when the gyroscope saturates, thus improving the robustness of downstream SLAM algorithms.

Inertial-only Odometry (IOO). Naive integration of IMU measurements — without aiding sources such as vision— typically leads to quick divergence of the odometric estimate. This is a cause of concern even in aided inertial odometry when the source of aiding becomes unavailable. For example, for hand tracking in mobile AR/VR applications, highly dynamic hands can easily move out of the tracking camera's FOV, leaving only IMU data available to keep motion tracking alive; or textureless scenes may prevent feature detection and tracking, causing VIO to only rely on IMU data. For this reason, recent work investigates the use of learning and neural networks to reduce the drift in inertial-only odometry [1211, 185, 1060, 459, 460, 228, 905]. These include attempts to model IMU bias in a data-driven manner with neural networks [233] or directly predicting displacements from a sequence of noisy IMU measurements [689]. For instance, one may use a differentiable integration module to integrate IMU readings with the predicted bias removed [1271, 905], or directly use ground truth bias for supervision [129], or use a conditional diffusion model to approximate bias which is modeled as a probability distribution [1293]. These methods have demonstrated the possibility of largely reducing the drift in inertial-only odometry, but currently they have limited generalization (*e.g.,* to different sensors or to motions not seen at training time).

Ultra-efficient and Robust VIO at the Edge. Despite recent advancements in SLAM, computational constraints arising in embedded robotic systems still pose critical challenges. Building robust VIO on these small form-factor platforms is hard due to strict size, weight, and power (SWAP) constraints, with the primary difficulty often arising from data management rather than computation. For example, in SLAM and hand-tracking modules of Meta XR wearable devices, the major energy consumption is data access in RAM [5]. To reduce the data transfer, an on-sensor computing architecture is presented [398], and a quantized visual-inertial odometry (QVIO) algorithms is developed in [872, 874]. For low-SWaP platforms, where only single-precision floating-point arithmetic is available on the computation unit or is required to speed up and achieve real-time performance, new square-root

(information or covariance) filters [873, 1191] have been introduced to improve efficiency while maintaining numerical stability. An ASIC design and implementation for on-chip visual-inertial odometry system is presented in [1274, 1053].