Inertial Sensor-Based Simultaneous Localization and Mapping for UAVs

21

Mitch Bryson and Salah Sukkarieh

Contents

M. Bryson (✉)
Australian Centre for Field Robotics, The University of Sydney, Sydney, NSW, Australia
e-mail: m.bryson@acfr.usyd.edu.au

S. Sukkarieh
Australian Centre for Field Robotics (ACFR), School of Aerospace, Mechanical & Mechatronic Engineering (AMME), The University of Sydney, Sydney, NSW, Australia
e-mail: salah@acfr.usyd.edu.au

K.P. Valavanis, G.J. Vachtsevanos (eds.), *Handbook of Unmanned Aerial Vehicles*,
DOI 10.1007/978-90-481-9707-1_5,

Abstract

This chapter provides an overview of algorithms for inertial sensor-based Simultaneous Localization and Mapping (SLAM) within the context of Unmanned Aerial Vehicles (UAVs). The presentation in this chapter is based on the use of the Extended Kalman Filter (EKF) and the Extended Information Filter (EIF) due to their ease of understanding, applicability to online implementation, and prevalence in airborne localization applications outside of SLAM (such as aided inertial localization). The discussion here includes an examination of SLAM for both small- and large-scale operation over the surface of the Earth, inertial SLAM using both range-bearing and bearing-only observations of the terrain, and a look at several different centralized and decentralized architectures for performing multi-vehicle SLAM.

21.1 Introduction

Simultaneous Localization and Mapping (SLAM) is the process of determining the position and orientation of a moving platform within an environment, while also building a map of the environment, primarily using observations of environmental features measured from and with respect to the moving platform. Since the seminal work by Smith et al. (1990), there have been several demonstrated implementations of SLAM using land (Bosse et al. 2003; Gutmann and Konolige 1999; Dissanayake et al. 2001; Guivant and Nebot 2001; Thrun and Lui 2003b) and underwater vehicles (Williams et al. 2001; Olson et al. 2004) where two-dimensional, horizontal localization and mapping is performed. In each of these implementations, SLAM involves the fusion of two particular sets of information usually coming from sensors onboard the vehicle; firstly the vehicle uses exteroceptive sensors to sense the terrain/landmarks relative to the vehicle (such as vision, laser, and radar sensors), and secondly the vehicle uses proprioceptive sensors and information to sense its own motion (such as wheel encoders, inertial sensors, and vehicle model information such as holonomic/nonholonomic constraints).

In the context of Unmanned Aerial Vehicles (UAVs), where the platform undergoes rapid six degree-of-freedom (6-DoF) motion, inertial sensors are a logical choice for the proprioceptive core of a SLAM implementation, due to their high frequency of information and ability to track platform motion in 6-DoF without relying on external information such as a vehicle dynamic model. Exteroceptive sensors (i.e., terrain sensors) come in a variety of forms depending on the application, including RAdio Detection And Ranging (RADAR), LIght Detection And Ranging (LIDAR), and electro-optical sensors such as cameras. Such sensors provide a variety of different observations of the terrain including range and bearing to features and bearing-only observations of features (i.e., camera images), necessitating SLAM algorithms that account for this variability.

The utility of SLAM in the context of UAV operations can be considered from both a localization and a mapping perspective. Traditionally, airborne localization has relied on external navigation aids such as satellite/radio positioning

systems (i.e., the Global Positioning System (GPS)) or terrain/landmark maps (i.e., in terrain-aided navigation systems (TANS)). SLAM allows for intermittent localization when external navigation aids fail due to, for example, satellite signal occlusion/jamming or when operating over unknown terrain, thus improving localization system robustness. During mapping and surveying tasks, a UAV uses terrain observations (e.g., aerial photography) along with information about the UAV's own position and orientation at the time of the observation to construct a map. Consistent and accurate mapping relies on accounting for the correlation between the errors in multiple features in the map, induced by a common error in the estimate of the platform's position and orientation; SLAM maintains these relationships, computing a joint estimate of the localization and mapping states, improving accuracy in the platform's localization estimates and thus in the final map. Large-scale mapping tasks may also necessitate the use of multiple aerial platforms; algorithms for multi-vehicle inertial SLAM are a natural extension of the single-vehicle case where shared map information can be used to assist in localization across platforms and to build more accurate maps than achievable with a single UAV.

This chapter provides an overview of algorithms for inertial sensor-based SLAM within the context of UAVs. The presentation in this chapter is based on the use of the Extended Kalman Filter (EKF) and the Extended Information Filter (EIF) due to their ease of understanding, applicability to online implementation, and prevalence in airborne localization applications outside of SLAM (i.e., aided inertial localization, see Giovanni 1979; Bar-Itzhack et al. 1982; Meyer-Hilberg and Jacob 1994). The discussion here includes an examination of SLAM for both small- and large-scale operation over the surface of the Earth, inertial SLAM using both range-bearing and bearing-only observations of the terrain and a look at several different centralized and decentralized architectures for performing multi-vehicle SLAM. Section 21.2 provides an overview of the equations of motion and sensor models used in inertial SLAM with an examination of the issue of SLAM in both global and local coordinate systems and how this affects the way in which inertial SLAM is implemented. Section 21.3 examines the structure of the inertial SLAM algorithm when terrain observations are made using a range/bearing sensor. Section 21.4 examines a special case of inertial SLAM when only bearing observations of the terrain are available, such as in the case of electro-optical sensing. Section 21.5 examines the inertial SLAM algorithm when applied to multiple vehicles where vehicles share map information with each other.

21.2 Inertial SLAM Sensor Equations

The inertial SLAM algorithm is formulated using an EKF and uses information from inertial sensors and from feature observations from onboard terrain sensors in order to estimate the vehicle's position, velocity, attitude, inertial sensor biases, and map feature locations. The algorithm works on a two-stage process of state prediction (where equations for predicting the vehicle states from inertial sensor data are used)

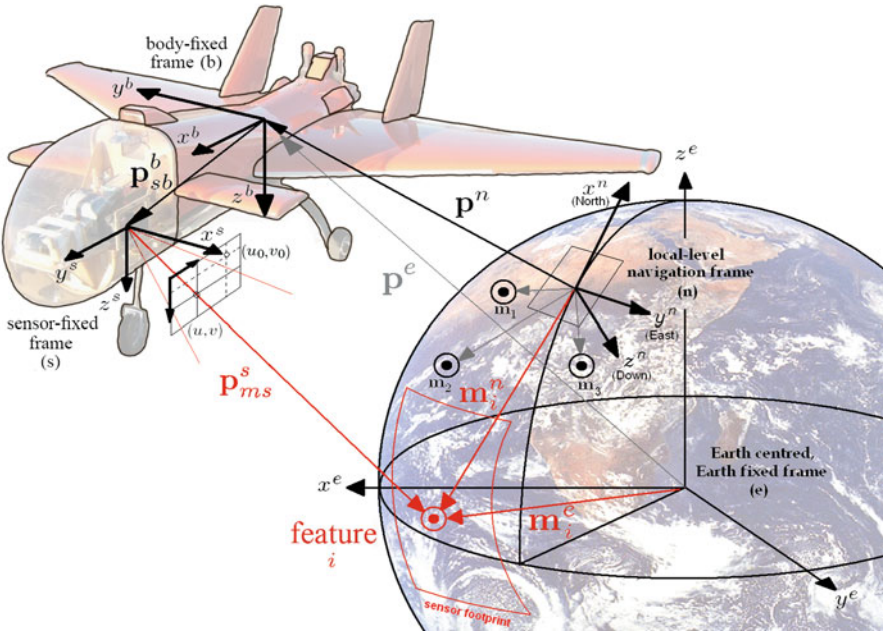

Fig. 21.1 Vectors and frames of reference in the inertial SLAM algorithm. The Earth-Centered, Earth-Fixed (ECEF) frame (e), local-level navigation frame (n), body-fixed frame (b), and sensor-fixed frame (s) and the relationship between the vehicle position (p^n,p^e), feature position (m^n,m^e), and sensor observation are shown. Also shown is the observation of a terrain feature coming from an example terrain sensor (in this case a vision camera)

and state observation/update (where equations describing the geometry of terrain feature observations are used).

This section describes the background equations behind inertial sensor-based SLAM algorithms, including equations of motion for inertial localization and sensor equations for terrain-observing sensors. Figure 21.1 illustrates the relevant relationships between the frames of reference and vectors used in the SLAM algorithm. The structure of the process and observation model equations for both global and local SLAM is discussed in the following subsections.

21.2.1 Global Vs. Local SLAM

Localization and mapping tasks vary in environmental scale depending on the application. Some tasks require knowledge of the location of each feature in the map with respect to global co-ordinates (such as an ECEF coordinate system (see Fig. 21.1)). This form of localization and mapping requires external information either in the form of global position information (such as from GPS) or global map information (i.e., the global position of one or more features in the map). Other

tasks may only require a map of the environment in which the positions of features are known only with respect to one another or with respect to the starting location of the vehicle, a task which when using SLAM does not require external map or navigation aids.

These two forms of SLAM are referred to as "global SLAM" or "local SLAM" depending on whether map and location information must be referenced to a global coordinate system or merely and local arbitrary coordinate system (such as the starting position of the UAV). This issue is particularly important when using inertial sensors as inertial localization relies to some degree on global information about the local coordinate system's motion with the rotation of the Earth. Local SLAM using inertial sensors is therefore only possible when certain assumptions are made which treat the local map coordinates as an inertial frame of reference by ignoring the effect of the Earth's rotation.

21.2.2 Inertial Localization and SLAM Process Model Equations

The presentation here begins by considering the process model equations from a global SLAM perspective. In this case, the location of the vehicle and the terrain is estimated with respect to the center of the Earth. The estimated state vector $\hat{\mathbf{x}}(k)$ at time step k in this case contains the three-dimensional vehicle position ($\mathbf{p}^e$), velocity ($\mathbf{v}^e$) and Euler angles ($\boldsymbol{\Psi}^n = [\phi, \theta, \psi]^T$), IMU sensor biases ($\delta\mathbf{f}^b$ and $\delta\omega_{ib}^b$), and the N three-dimensional feature locations ($\mathbf{m}_i^e$) in the environment:

$$\hat{\mathbf{x}}(k) = [\mathbf{p}^e(k), \mathbf{v}^e(k), \boldsymbol{\Psi}^n(k), \delta\mathbf{f}^b(k), \delta\omega_{ib}^b(k), \mathbf{m}_1^e(k), \mathbf{m}_2^e(k), \ldots, \mathbf{m}_N^e(k)]^T \tag{21.1}$$

where $i = 1, \ldots, N$, the superscript e indicates Earth-Centered, Earth-Fixed (ECEF) frame referenced vectors, the superscript b indicates body-fixed frame referenced vectors and the superscript n indicates the Euler angles $\boldsymbol{\Psi}^n$ parameterize the body-fixed to local navigation frame Direction Cosine Matrix (DCM) transformation $\mathbf{C}_b^n$ (see Chapter 3 of this book for details on frames of reference). Figure 21.1 illustrates the relevant relationships between the frames of reference and vectors used in the SLAM algorithm.

Euler angles are used to represent the attitude of the platform rather than quaternions in order to reduce the number of parameters in the estimator. It is assumed that the vehicle's pitch angle is constrained so as to avoid the Euler angle singularity at $\theta = 90°$; however, the process model could be adapted to implement quaternions for the parameterization of $\mathbf{C}_b^n$ in the event that the platform motion is not constrained in such a way.

The state estimate $\hat{\mathbf{x}}$ is predicted forward in time by integrating the first-order nonlinear dynamic equation:

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{F}_c[\hat{\mathbf{x}}(t), \mathbf{u}(t)] + \mathbf{G}_c[\mathbf{w}(t)] \tag{21.2}$$

where $\mathbf{F}_c[.,.]$ is the continuous-time process model function, $\mathbf{G}_c[.]$ is the continuous-time input function, $\mathbf{u}(t)$ is the system input (which is comprised of inertial sensor readings), and $\mathbf{w}(t)$ is uncorrelated, zero-mean vehicle process noise vector of covariance $\mathbf{Q}$ (which is comprised of inertial sensor noise errors). The process model equations for the vehicle position, velocity, and attitude are based on the 6-DoF inertial localization equations in which an Earth-frame mechanization (Titterton and Weston 1997) is applied:

$$\dot{\mathbf{p}}^e = \mathbf{v}^e \tag{21.3}$$

$$\dot{\mathbf{v}}^e = \mathbf{C}_b^e \hat{\mathbf{f}}^b - \mathbf{C}_b^e \delta \mathbf{f}^b - \mathbf{C}_b^e \mathbf{w}_{\text{accel}} - 2(\omega_{ie}^e \times \mathbf{v}^e) + \mathbf{g}_l^e \tag{21.4}$$

$$\dot{\boldsymbol{\Psi}}^n = \mathbf{E}_b^n (\hat{\omega}_{ib}^b - \delta \omega_{ib}^b - \mathbf{w}_{\text{gyro}} - \mathbf{C}_n^b \omega_{ie}^n) \tag{21.5}$$

where ω_{ie}^e and ω_{ie}^n are the Earth's rotation rate vectors measured in the ECEF and local navigation frames, respectively:

$$\omega_{ie}^e = \begin{bmatrix} 0 \\ 0 \\ \omega_{\text{Earth}} \end{bmatrix} \tag{21.6}$$

$$\omega_{ie}^n = \begin{bmatrix} \omega_{\text{Earth}} \cos(\lambda) \\ 0 \\ -\omega_{\text{Earth}} \sin(\lambda) \end{bmatrix} \tag{21.7}$$

where λ is the latitude angle of the vehicle and $\omega_{\text{Earth}} = 7.292115 \times 10^{-5}$ rad/s is the rotation rate of the Earth. $\mathbf{E}_b^n$ is the body to navigation frame rotation rate transformation matrix:

$$\mathbf{E}_b^n = \begin{bmatrix} 1 & \sin\phi \tan\theta & \cos\phi \tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi \sec\theta & \cos\phi \sec\theta \end{bmatrix} \tag{21.8}$$

$\mathbf{g}_l^e$ is the local gravity term:

$$\mathbf{g}_l^e = \mathbf{g}^e - \omega_{ie}^e \times (\omega_{ie}^e \times \mathbf{p}^e) \tag{21.9}$$

$$= -g \begin{bmatrix} \frac{\mathbf{p}_x^e}{|\mathbf{p}^e|} \\ \frac{\mathbf{p}_y^e}{|\mathbf{p}^e|} \\ \frac{\mathbf{p}_z^e}{|\mathbf{p}^e|} \end{bmatrix} - [\times \omega_{ie}^e]^2 \mathbf{p}^e \tag{21.10}$$

where g is the acceleration acting on the vehicle due to gravity, referenced in ECEF coordinates and $[\times \omega_{ie}^e]$ is the skew symmetric matrix of the Earth's rotation rate vector. $\mathbf{C}_b^n$ is the DCM transformation from the body to local navigation frame, and $\mathbf{C}_b^e$ is the DCM transformation from the body to ECEF frame which are related via

$$\mathbf{C}_b^e = \mathbf{C}_n^e \mathbf{C}_b^n \tag{21.11}$$

$$C_n^e = \begin{bmatrix} -\sin(\lambda)\cos(l) & -\sin(l) & -\cos(\lambda)\cos(l) \\ -\sin(\lambda)\sin(l) & \cos(l) & -\cos(\lambda)\sin(l) \\ \cos(\lambda) & 0 & -\sin(\lambda) \end{bmatrix} \tag{21.12}$$

where l is the longitude angle of the vehicle's position. The vectors $\hat{\mathbf{f}}^b$ and $\hat{\omega}_{ib}^b$ are the accelerometer specific force vector reading and gyroscope rotation rate reading, respectively (where $\mathbf{u}(t) = [\hat{\mathbf{f}}^b(t), \hat{\omega}_{ib}^b(t)]^T$), $\delta\mathbf{f}^b$ and $\delta\omega_{ib}^b$ are the accelerometer and gyro biases, respectively, and $\mathbf{w}_{\text{accel}}$ and $\mathbf{w}_{\text{gyro}}$ are the accelerometer and gyro noise values. The true acceleration and rotation rates, $\mathbf{f}^b$ and ω_{ib}^b, are defined:

$$\mathbf{f}^b = \hat{\mathbf{f}}^b - \delta\mathbf{f}^b - \mathbf{w}_{\text{accel}} \tag{21.13}$$

$$\omega_{ib}^b = \hat{\omega}_{ib}^b - \delta\omega_{ib}^b - \mathbf{w}_{\text{gyro}} \tag{21.14}$$

Accelerometer and gyro biases are known to fluctuate by very small amounts due to temperature changes and bias wander, particularly in low-cost IMUs. The accelerometer and gyro bias process models are given by

$$\delta\dot{\mathbf{f}}^b = \mathbf{w}_{b,\text{accel}} \tag{21.15}$$

$$\delta\dot{\omega}_{ib}^b = \mathbf{w}_{b,\text{accel}} \tag{21.16}$$

where $\mathbf{w}_{b,\text{accel}}$ and $\mathbf{w}_{b,\text{gyro}}$ are bias drift noises for the accelerometers and gyros, respectively. The complete noise vector $\mathbf{w}(t)$ for the process model based on errors from the IMU is composed:

$$\mathbf{w}(t) = [\mathbf{w}_{\text{accel}}(t), \mathbf{w}_{\text{gyro}}(t), \mathbf{w}_{b,\text{accel}}(t), \mathbf{w}_{b,\text{gyro}}(t)]^T \tag{21.17}$$

Finally, map feature locations are estimated in the ECEF frame and are also assumed to be constant as the state of the terrain is stationary and the process model of the ith feature is given by

$$\dot{\mathbf{m}}_i^e = \mathbf{0} \tag{21.18}$$

Equations 21.3–21.5, 21.15, 21.16, and 21.18 can also be expressed in discrete-time, recursive form by assuming a first-order Euler integration step:

$$\hat{\mathbf{x}}(k) = \mathbf{F}[\hat{\mathbf{x}}(k-1), \mathbf{u}(k), k] + \mathbf{G}[\mathbf{w}(k)] \tag{21.19}$$

where $\mathbf{F}[.,.,k]$ is the nonlinear state transition function at time k and $\mathbf{G}[.,k]$ is the input model transition function at time k. The discrete process model thus becomes

$$\mathbf{p}^e(k) = \mathbf{p}^e(k-1) + \mathbf{v}^e \Delta t \tag{21.20}$$

$$\mathbf{v}^e(k) = \mathbf{v}^e(k-1) + [\mathbf{C}_b^e \hat{\mathbf{f}}^b - \mathbf{C}_b^e \delta\mathbf{f}^b - \mathbf{C}_b^e \mathbf{w}_{\text{accel}} - 2(\omega_{ie}^e \times \mathbf{v}^e) + \mathbf{g}_l^e]\Delta t \tag{21.21}$$

$$\boldsymbol{\Psi}^n(k) = \boldsymbol{\Psi}^n(k-1) + [\mathbf{E}_b^n(\hat{\omega}_{ib}^b - \delta\omega_{ib}^b - \mathbf{w}_{\text{gyro}} - \mathbf{C}_n^b \omega_{ie}^n)]\Delta t \tag{21.22}$$

$$\delta\mathbf{f}^b(k) = \delta\mathbf{f}^b(k-1) + \mathbf{w}_{b,\text{accel}}(k) \tag{21.23}$$

$$\delta\omega_{ib}^b(k) = \delta\omega_{ib}^b(k-1) + \mathbf{w}_{b,\text{gyro}}(k) \tag{21.24}$$

$$\mathbf{m}_i^e(k) = \mathbf{m}_i^e(k-1) \tag{21.25}$$

where Δt is the time difference between the k and $k-1$ discrete-time segments.

21.2.2.1 Process Model Equation Approximations for Local SLAM

When performing SLAM with respect to a local coordinate system where no global information is available, several approximations must be made in the inertial SLAM equations. In local SLAM, the position and orientation of the vehicle is maintained with respect to a local navigation frame which is fixed to an arbitrary and unknown location on the Earth's surface rather than the ECEF frame. The local SLAM equations are thus derived under the assumption that local navigation frame is an inertial frame of reference by ignoring the small Coriolis and centripetal accelerations which are incurred by the Earth's rotation.

In local SLAM, the estimated state vector is $\hat{\mathbf{x}}_{\mathbf{local}}(k)$, in which the vehicle position and velocity and the position of map features are now reference in local navigation frame coordinates:

$$\hat{\mathbf{x}}_{\mathbf{local}}(k) = [\mathbf{p}^n(k), \mathbf{v}^n(k), \boldsymbol{\Psi}^n(k), \delta\mathbf{f}^b(k), \delta\omega_{ib}^b(k), \mathbf{m}_1^n(k), \mathbf{m}_2^n(k), \ldots, \mathbf{m}_N^n(k)]^T \tag{21.26}$$

The continuous-time and discrete-time process models for local SLAM are thus given by

$$\dot{\hat{\mathbf{x}}}_{\text{local}}(t) = \mathbf{F}_{c,\text{local}}[\hat{\mathbf{x}}_{\text{local}}(t), \mathbf{u}(t)] + \mathbf{G}_{c,\text{local}}[\mathbf{w}(t)] \tag{21.27}$$

$$\hat{\mathbf{x}}_{\text{local}}(k) = \mathbf{F}_{\text{local}}[\hat{\mathbf{x}}_{\text{local}}(k-1), \mathbf{u}(k), k] + \mathbf{G}_{\text{local}}[\mathbf{w}(k)] \tag{21.28}$$

The vehicle process model equations in continuous-time form are the following:

$$\dot{\mathbf{p}}^n = \mathbf{v}^n \tag{21.29}$$

$$\dot{\mathbf{v}}^n = \mathbf{C}_b^n \hat{\mathbf{f}}^b - \mathbf{C}_b^n \delta\mathbf{f}^b - \mathbf{C}_b^n \mathbf{w}_{\text{accel}} + \mathbf{g}^n \tag{21.30}$$

$$\dot{\boldsymbol{\Psi}}^n = \mathbf{E}_b^n(\hat{\omega}_{ib}^b - \delta\omega_{ib}^b - \mathbf{w}_{\text{gyro}}) \tag{21.31}$$

where $\mathbf{g}^n = [0, 0, g]^T$ is the vector of acceleration due to gravity in the local navigation frame and $g = 9.81\,\text{m/s}^2$. Terrain map features are now referenced in local navigation frame coordinates, and their process model is given by

$$\dot{\mathbf{m}}_i^n = \mathbf{0} \tag{21.32}$$

The complete form of the discrete-time process model equations in local SLAM are thus

$$\mathbf{p}^n(k) = \mathbf{p}^n(k-1) + \mathbf{v}^n \Delta t \tag{21.33}$$

$$\mathbf{v}^n(k) = \mathbf{v}^n(k-1) + [\mathbf{C}_b^n \hat{\mathbf{f}}^b - \mathbf{C}_b^n \delta\mathbf{f}^b - \mathbf{C}_b^n \mathbf{w}_{\text{accel}} + \mathbf{g}^n]\Delta t \tag{21.34}$$

$$\boldsymbol{\Psi}^n(k) = \boldsymbol{\Psi}^n(k-1) + [\mathbf{E}_b^n(\hat{\omega}_{ib}^b - \delta\omega_{ib}^b - \mathbf{w}_{\text{gyro}})]\Delta t \tag{21.35}$$

$$\delta\mathbf{f}^b(k) = \delta\mathbf{f}^b(k-1) + \mathbf{w}_{b,\text{accel}}(k) \tag{21.36}$$

$$\delta\omega_{ib}^b(k) = \delta\omega_{ib}^b(k-1) + \mathbf{w}_{b,\text{gyro}}(k) \tag{21.37}$$

$$\mathbf{m}_i^n(k) = \mathbf{m}_i^n(k-1) \tag{21.38}$$

21.2.3 Landmark/Terrain Sensor Equations

The observation model equations describe the relationship between the sensor observation of a map feature in the sensor coordinates of the terrain sensor to the estimated states in SLAM. Figure 21.1 illustrates the relevant relationships between the sensor frame, sensor observation and map, and vehicle positions. The observation $\mathbf{z}_i(k)$ is related to the estimated states using Eq. 21.39 for global SLAM and Eq. 21.40 for local SLAM:

$$\mathbf{z}_i(k) = \mathbf{H}_i(\mathbf{p}^e(k), \boldsymbol{\Psi}^n(k), \mathbf{m}_i^e(k), k) + \mathbf{v}(k) \tag{21.39}$$

$$\mathbf{z}_i(k) = \mathbf{H}_{i,\text{local}}(\mathbf{p}^n(k), \boldsymbol{\Psi}^n(k), \mathbf{m}_i^n(k), k) + \mathbf{v}(k) \tag{21.40}$$

where $\mathbf{H}_i(.,.,.,k)$ and $\mathbf{H}_{i,\text{local}}(.,.,.,k)$ are functions of the feature location, vehicle position, and Euler angles and $\mathbf{v}(k)$ is uncorrelated, zero-mean observation noise errors of covariance $\mathbf{R}$.

Terrain observations could come from a variety of different sensors such as RADAR, LIDAR, or an electro-optical camera. The SLAM algorithm requires that point features can be extracted from the observation sensor data which go on to be mapped terrain features. In the case of large objects that do not show up as a "point" in sensor data, the centroid of the object is found or else several points can be used to represent a single object. Example feature extraction algorithms for vision include Scale-Invarient Feature Transform (SIFT) features (Lowe 2004) or model-based feature matching (Nixon and Aguado 2001). Features in this sense are points in the sensor data that are distinct and easily recognizable or else points in

the sensor data that appear to correlate well with a given feature model or template that is specified offline.

Sensor observations can be broken up into two types: firstly range/bearing observations (as might be available from RADAR or a LIDAR) and secondly bearing-only observations (as might be available from an electro-optical sensor such as a camera). When both range and bearing observations are made, the observation model is given by

$$\mathbf{z}_i(k) = \begin{bmatrix} \rho_i \\ \varphi_i \\ \vartheta_i \end{bmatrix} = \begin{bmatrix} \sqrt{(x_i^s)^2 + (y_i^s)^2 + (z_i^s)^2} \\ \tan^{-1}\left(\frac{y_i^s}{x_i^s}\right) \\ \tan^{-1}\left(\frac{z_i^s}{\sqrt{(x_i^s)^2+(y_i^s)^2}}\right) \end{bmatrix} \tag{21.41}$$

where ρ_i, φ_i, and ϑ_i are the observed range, azimuth, and elevation angles to the feature and x_i^s, y_i^s, and z_i^s are the Cartesian coordinates of $\mathbf{p}_{ms,i}^s$, the relative position of the ith feature with respect to the sensor, measured in the sensor frame.

There are two forms that can be used to represent a bearing-only observation. The observation $\mathbf{z}_i(k)$ can be represented by azimuth (φ_i) and elevation angles (ϑ_i):

$$\mathbf{z}_{i,\text{ang}}(k) = \begin{bmatrix} \varphi_i \\ \vartheta_i \end{bmatrix} = \begin{bmatrix} \tan^{-1}\left(\frac{y_i^s}{x_i^s}\right) \\ \tan^{-1}\left(\frac{z_i^s}{\sqrt{(x_i^s)^2+(y_i^s)^2}}\right) \end{bmatrix} \tag{21.42}$$

where $\mathbf{R}_{\text{ang}}$ is the angular noise covariance. For vision camera, the observation is better represented as pixels in the image of the camera, using a pinhole camera model:

$$\mathbf{z}_{i,\text{pix}}(k) = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_u(\frac{y_i^s}{x_i^s}) + u_0 \\ f_v(\frac{z_i^s}{x_i^s}) + v_0 \end{bmatrix} \tag{21.43}$$

where u_0, v_0, f_u, and f_v are calibration parameters for the camera and the pixel noise covariance is $\mathbf{R}_{\text{pix}}$. The relationship between the pixel coordinates and the azimuth and elevation angles is given by Eq. 21.44

$$\begin{bmatrix} \varphi \\ \vartheta \end{bmatrix} = \begin{bmatrix} \tan^{-1}\frac{(u-u_0)}{f_u} \\ \tan^{-1}(\frac{(v-v_0)}{f_v}\cos\varphi) \end{bmatrix} \tag{21.44}$$

$\mathbf{p}_{ms,i}^s$, the relative position of the ith feature with respect to the sensor, measured in the sensor frame is given by:

$$\begin{aligned} \mathbf{p}_{ms,i}^s &= \mathbf{C}_b^s \mathbf{C}_n^b [\mathbf{m}_i^n - \mathbf{p}^n - \mathbf{C}_b^n \mathbf{p}_{sb}^b] \\ &= \mathbf{C}_b^s \mathbf{C}_e^b [\mathbf{m}_i^e - \mathbf{p}^e - \mathbf{C}_b^e \mathbf{p}_{sb}^b] \end{aligned} \tag{21.45}$$

where $\mathbf{C}_b^s$ is the transformation matrix from the body frame to the sensor frame and $\mathbf{p}_{sb}^b$ is the sensor offset from the vehicle center of mass, measured in the body frame, otherwise known as the "lever arm" and $\mathbf{C}_e^b = (\mathbf{C}_b^e)^T$.

21.3 Inertial SLAM with Range and Bearing Sensors

When both range and bearing observations to features are available, the inertial SLAM algorithm is broken up into the following processes:

1. *State Prediction.* The process model equations in Sect. 21.2.2 are used to "predict" forward in time the estimated state vector and state covariance matrix in an EKF prediction step.
2. *Data Association.* Every time a new feature observation is made, the data association step is used to determine whether the observation is of a new feature or a known feature, and which feature it is of.
3. *Feature Initialization.* When a feature is observed by the terrain sensor for the first time, its position is computed and augmented into the estimated state vector and its initial position uncertainty augmented into the state covariance matrix.
4. *State Update.* Each time an observation is made of a previously seen feature, it is used in an EKF update step to correct the value of the estimated state and update the state covariance matrix.

Figure 21.2 provides an overview of the range/bearing inertial SLAM algorithm. At each time step, inertial sensor data is used for the prediction stage. When features are extracted from the sensor data, they are run through a data association stage. New features are augmented into the state vector and state covariance matrix, and observations of existing features are used to update the estimated state vector and state covariance matrix. The following subsections summarize the equations and methods in each step of the algorithm. For further details of inertial SLAM with range and bearing sensors and an implementation example, the reader is referred to Kim and Sukkarieh (2003).

21.3.1 State Prediction

The state prediction stage is run recursively each time a new reading is taken from the inertial sensors. In the case of global SLAM, the estimated state vector $\hat{\mathbf{x}}(k+1)$ is predicted from the previous time step state estimate using Eqs. 21.20–21.25. The state covariance $\mathbf{P}^-(k+1)$, which is the state covariance at time step k after prediction, is computed as

$$\mathbf{P}^-(k+1) = \nabla\mathbf{F}\mathbf{P}(k)\nabla\mathbf{F}^T + \nabla\mathbf{G}\mathbf{Q}(k)\nabla\mathbf{G}^T \tag{21.46}$$

where $\nabla\mathbf{F}$ and $\nabla\mathbf{G}$ are the Jacobians of the state transition function $\mathbf{F}[.,.,k]$ with respect to the state vector $\hat{\mathbf{x}}(k+1)$ and input model transition function $\mathbf{G}[.,k]$ with

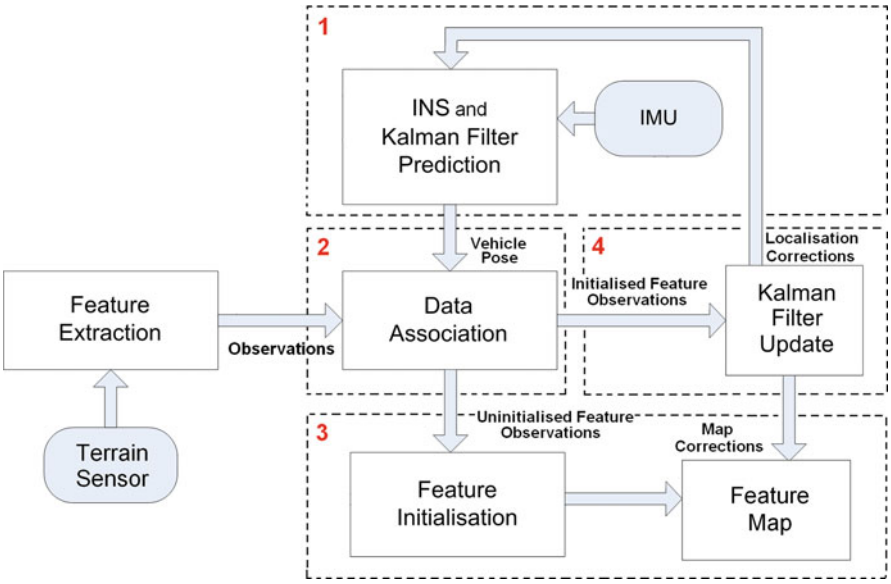

Fig. 21.2 Overview of the range/bearing inertial SLAM algorithm. the algorithm is recursive; EKF state prediction (*1*) is performed when inertial sensor data is available. Data association (*2*), feature initialization (*3*), and EKF state update (*4*) are performed when terrain sensor observations are made

respect to the noise input $\mathbf{w}(k+1)$, respectively. In the case of local SLAM, the estimated state vector $\hat{\mathbf{x}}_{\text{local}}(k)$ is predicted forward using Eqs. 21.33–21.38. The state covariance $\mathbf{P}^{-}(k+1)$ is computed as

$$\mathbf{P}^{-}(k+1) = \nabla\mathbf{F}_{\text{local}}\mathbf{P}(k)\nabla\mathbf{F}_{\text{local}}^{T} + \nabla\mathbf{G}_{\text{local}}\mathbf{Q}(k)\nabla\mathbf{G}_{\text{local}}^{T} \tag{21.47}$$

where $\nabla\mathbf{F}_{\text{local}}$ and $\nabla\mathbf{G}_{\text{local}}$ are the Jacobians of the state transition function $\mathbf{F}_{\text{local}}[.,.,k]$ with respect to the state vector $\hat{\mathbf{x}}_{\text{local}}(k+1)$ and input model transition function $\mathbf{G}_{\text{local}}[.,k]$ with respect to the noise input $\mathbf{w}(k+1)$, respectively.

21.3.2 Feature Initialization

When the first range/bearing observation of a particular feature is obtained, its position is calculated using the initialization function $\mathbf{J}_1[\hat{\mathbf{x}}(k), \mathbf{J}_2(\mathbf{z}_i(k))]$ which is given as

$$\mathbf{J}_1[\hat{\mathbf{x}}(k), \mathbf{J}_2(\mathbf{z}_i(k))] \longrightarrow \mathbf{m}_i^e = \mathbf{p}^e + \mathbf{C}_n^e\mathbf{C}_b^n\mathbf{p}_{sb}^b + \mathbf{C}_n^e\mathbf{C}_b^n\mathbf{C}_s^b\mathbf{p}_{ms}^s \tag{21.48}$$

for the case of global SLAM and

$$\mathbf{J}_{1,\text{local}}[\hat{\mathbf{x}}_{\text{local}}(k), \mathbf{J}_2(\mathbf{z}_i(k))] \longrightarrow \mathbf{m}_i^n = \mathbf{p}^n + \mathbf{C}_b^n\mathbf{p}_{sb}^b + \mathbf{C}_b^n\mathbf{C}_s^b\mathbf{p}_{ms}^s \tag{21.49}$$

for the case of local SLAM, where

$$\mathbf{J}_2(\mathbf{z}_i(k)) \longrightarrow \mathbf{p}^s_{ms,i} = \begin{bmatrix} \rho_i \cos(\varphi_i)\cos(\vartheta_i) \\ \rho_i \sin(\varphi_i)\cos(\vartheta_i) \\ \rho_i \sin(\vartheta_i) \end{bmatrix} \tag{21.50}$$

for both the global and local SLAM cases. The state vector and covariance are then augmented to include the new feature position:

$$\hat{\mathbf{x}}_{\text{aug}}(k) = \begin{bmatrix} \hat{\mathbf{x}}(k) \\ \mathbf{m}^e_i(k) \end{bmatrix} \tag{21.51}$$

$$\mathbf{P}_{\text{aug}}(k) = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \nabla\mathbf{J}_x & \nabla\mathbf{J}_z \end{bmatrix} \begin{bmatrix} \mathbf{P}(k) & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_k \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \nabla\mathbf{J}_x & \nabla\mathbf{J}_z \end{bmatrix}^T \tag{21.52}$$

for the case of global SLAM and

$$\hat{\mathbf{x}}_{\text{local,aug}}(k) = \begin{bmatrix} \hat{\mathbf{x}}_{\text{local}}(k) \\ \mathbf{m}^n_i(k) \end{bmatrix} \tag{21.53}$$

$$\mathbf{P}_{\text{local,aug}}(k) = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \nabla\mathbf{J}_x & \nabla\mathbf{J}_z \end{bmatrix} \begin{bmatrix} \mathbf{P}_{\text{local}}(k) & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_k \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \nabla\mathbf{J}_x & \nabla\mathbf{J}_z \end{bmatrix}^T \tag{21.54}$$

for the case of local SLAM where $\nabla\mathbf{J}_x$ and $\nabla\mathbf{J}_z$ are the Jacobians of the initialization function $\mathbf{J}_1$ with respect to the state estimate $\hat{\mathbf{x}}(k)$ and the observation $\mathbf{z}_i(k)$, respectively. The position of this feature becomes correlated to all of the vehicle states including position, velocity, and attitude along with inertial sensor biases and also to the position of other features in the map.

21.3.3 Data Association

Data association is the process of matching observations of features from the terrain sensor with the estimated 3D position of the feature within the map. The validity of potential associations between observations and features is assessed using the Mahalanobis distance (γ) (Neira and Tardos 2001) in the sensor space (range, azimuth, and elevation):

$$\gamma_i = \nu_i(k)^T \mathbf{S}_i(k)^{-1} \nu_i(k) \tag{21.55}$$

where $\nu_i(k)$ and $\mathbf{S}_i(k)$ are the innovation and innovation covariance (see Sect. 21.3.4) for an observation of the ith feature in the map at time segment k.

When checking the association of a given observation with the initialized features in the map, γ_i is calculated for each initialized feature. Matchings that fall within a defined threshold of γ_i corresponding to a 95 % level of confidence are considered

acceptable, in which case the observation is associated to a known feature in the map and an EKF state update is performed (see Sect. 21.3.4 below). If the observation does not fall within the threshold of an existing map feature, it is assumed that the feature has not been seen before, and thus the observation data is used to initialize the feature into the map as is shown in Sect. 21.3.2.

This method of data association works well when feature observations are well spaced in the sensor coordinates with respect to the uncertainty of the vehicle position and orientation. In the case of very dense feature observations, it may be necessary to consider not only matches of individual features to individual sensor observations but also the joint compatibility of several features to several observations simultaneously as to overcome association ambiguity. For efficient techniques for joint compatibility data association based on the innovation gate in Eq. 21.55, the reader is referred to Neira and Tardos (2001).

21.3.4 State Update

Once a feature has been initialized into the state vector, subsequent observations of this feature are used to update the entire state vector consisting of the vehicle pose, velocity, inertial sensor biases, and the position of this feature and other features in the environment. The state estimate is updated in an EKF update step where the updated state estimate and state covariance matrix after the update are as follows:

$$\hat{\mathbf{x}}^{+}(k+1) = \hat{\mathbf{x}}^{-}(k+1) + \mathbf{W}(k+1)\nu(k+1) \tag{21.56}$$

$$\mathbf{P}^{+}(k+1) = \mathbf{P}^{-}(k+1) - \mathbf{W}(k+1)\mathbf{S}_i(k+1)\mathbf{W}(k+1)^T \tag{21.57}$$

$$\nu_i(k+1) = \mathbf{z}_i - \mathbf{H}_i(\hat{\mathbf{x}}^{-}(k+1)) \tag{21.58}$$

$$\mathbf{S}_i(k+1) = \nabla\mathbf{H}_i\mathbf{P}^{-}(k+1)\nabla\mathbf{H}_i^T + \mathbf{R}(k+1) \tag{21.59}$$

$$\mathbf{W}(k+1) = \mathbf{P}^{-}(k+1)\nabla\mathbf{H}_i^T\mathbf{S}_i^{-1}(k+1) \tag{21.60}$$

for the case of global SLAM and

$$\hat{\mathbf{x}}^{+}_{\text{local}}(k+1) = \hat{\mathbf{x}}^{-}_{\text{local}}(k+1) + \mathbf{W}(k+1)\nu(k+1) \tag{21.61}$$

$$\mathbf{P}^{+}_{\text{local}}(k+1) = \mathbf{P}^{-}_{\text{local}}(k+1) - \mathbf{W}(k+1)\mathbf{S}_i(k+1)\mathbf{W}(k+1)^T \tag{21.62}$$

$$\nu_i(k+1) = \mathbf{z}_i - \mathbf{H}_{i,\text{local}}(\hat{\mathbf{x}}^{-}_{\text{local}}(k+1)) \tag{21.63}$$

$$\mathbf{S}_i(k+1) = \nabla\mathbf{H}_{i,\text{local}}\mathbf{P}^{-}_{\text{local}}(k+1)\nabla\mathbf{H}_{i,\text{local}}^T + \mathbf{R}(k+1) \tag{21.64}$$

$$\mathbf{W}(k+1) = \mathbf{P}^{-}_{\text{local}}(k+1)\nabla\mathbf{H}_{i,\text{local}}^T\mathbf{S}_i^{-1}(k+1) \tag{21.65}$$

for the case of local SLAM where $\mathbf{H}_i(\hat{\mathbf{x}}^{-}(k+1))$ or $\mathbf{H}_{i,\text{local}}(\hat{\mathbf{x}}^{-}_{\mathbf{local}}(k+1))$ is the predicted feature observation which is computed from the estimated vehicle

position and attitude and estimate map location using Eqs. 21.41 and 21.45. $\nabla \mathbf{H}_i$ and $\nabla \mathbf{H}_{i,\text{local}}$ are the Jacobians of the observation function with respect to the predicted state vector. Once a feature leaves the field of view of the sensor, its position remains in the state vector and continues to be updated via its correlations to other visible features in the state vector.

21.4 Inertial SLAM with Bearing-Only Sensors

When bearing-only observations are made using a terrain sensor such as in the case of a vision system, additional elements must be added to the SLAM algorithm. Performing SLAM with bearing-only observations poses two main additional challenges to the range and bearing case. Firstly, a single bearing-only observation provides insufficient information alone to localize a feature in 3D. Instead observations from two sufficiently different poses are required. Secondly, data association is complicated by bearing-only observations. Since the 3D position of the feature is not known from a single observation, the Mahalanobis distance (Neira and Tardos 2001), commonly used for validation gating in tracking tasks, and as shown for the range-bearing case in Sect. 21.3.3, cannot be calculated in the standard way.

This section details variations to the inertial SLAM algorithms that account for complications arising from the use of bearing-only terrain sensors. The bearing-only inertial SLAM algorithm is broken up into the following processes:

1. *State Prediction.* The prediction step in the bearing-only case is performed in a similar manner as in the range/bearing case.
2. *Data Association.* Data association for initialized features follows the same methods as shown in the range/bearing case. Data association of uninitialized features is tackled by creating multi-hypothesis distributions of the possible feature locations in 3D (i.e., along the line of sight of an observation). Subsequent observations of the same feature can be associated by matching the most likely hypotheses and culling the hypotheses that do not match.
3. *Feature Initialization.* A delayed initialization technique is used to store information from bearing-only observations until there exists two observations with a sufficient baseline from which to initialize the 3D position of the feature. Once this is available, all of the information contained in stored observations is recovered.
4. *State Update.* The update step in the bearing-only case is performed in a similar manner as in the range/bearing case where bearing-only observations are used to update the estimated state which includes vehicle states, stored pose data, and features whose 3D positions have been initialized into the map.

Figure 21.3 provides an overview of the bearing-only inertial SLAM algorithm. The following subsections summarize the equations and methods in each step of the algorithm. For further details of inertial SLAM with bearing-only sensors and an implementation example, the reader is referred to Bryson and Sukkarieh (2007).

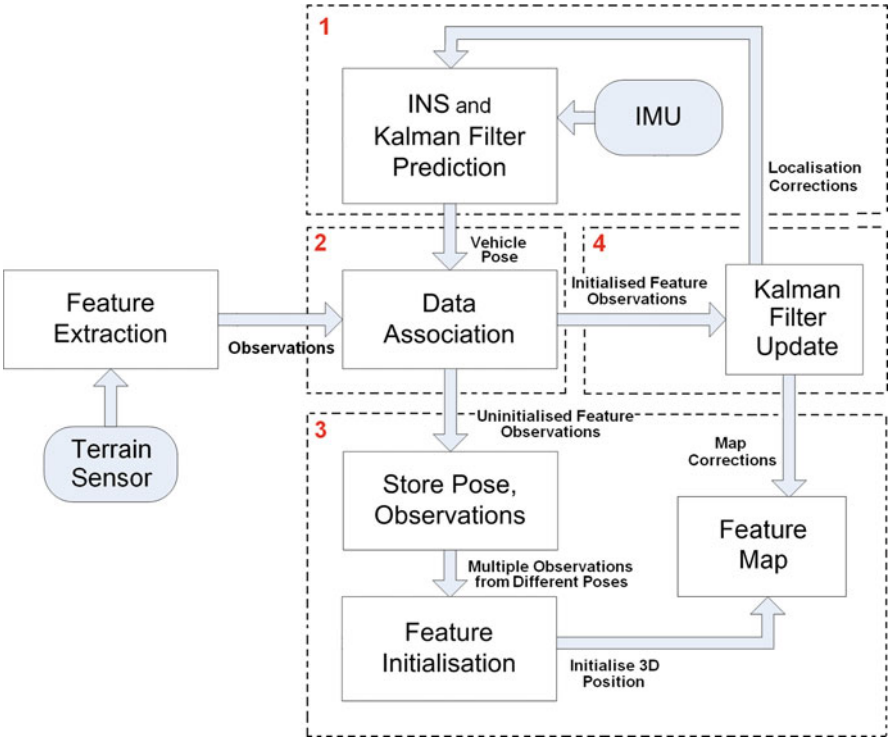

Fig. 21.3 Overview of the bearing-only inertial SLAM algorithm. The algorithm is recursive; EKF state prediction (*1*) is performed when inertial sensor data is available. Data association (*3*) and EKF state update (*4*) are performed when terrain sensor observations are made of features that have been already initialized into the map. Observations of new features are stored until enough observations exist to initialize the position of the feature into the state vector (*2*)

21.4.1 State Prediction and Update

The bearing-only SLAM equations rely on the storing of vehicle pose data into the estimated state vector. The methods for storing pose data is shown below in Sect. 21.4.2. The state prediction for the bearing-only case follows the same equations and methods as shown in the range/bearing observation case as shown in Sect. 21.3.1 except that stored pose data is predicted forward in the predict step in the same way as stored map features. The only difference in the state update between the bearing-only case and range/bearing case (shown in Sect. 21.3.4) is that now no range information is available in the observation. Equations 21.56–21.65 are used for the update where Eqs. 21.42 or 21.43 is used to compute $\mathbf{H}_i(\hat{\mathbf{x}}^-(k+1))$ or $\mathbf{H}_{i,\text{local}}(\hat{\mathbf{x}}^-_{\mathbf{local}}(k+1))$, the predicted feature observation, depending on the exact form of bearing observation from the terrain sensor (i.e., bearing angles or pixels).

21.4.2 Feature Initialization

A single bearing-only observation is insufficient to initialize the 3D position of a feature into the SLAM filter with Gaussian uncertainty. The following subsection outlines a method for delayed initialization of a feature into the filter by using stored observations and vehicle pose information. For simplification, the feature position is always initialized into the local navigation frame first before being transformed into the ECEF frame if global SLAM is performed. Otherwise, the feature position remains in the local navigation frame.

21.4.2.1 Storing Feature Observations and Vehicle Pose Information

When an observation of an uninitialized feature is made, the current bearing-only observation is stored, and the SLAM state vector is augmented to include the current vehicle pose (three position states and three Euler angle states):

$$\hat{\mathbf{x}}_v = \begin{bmatrix} \mathbf{p}^e(k) \\ \mathbf{v}^e(k) \\ \boldsymbol{\Psi}^n(k) \\ \delta\mathbf{f}^b(k) \\ \delta\omega_{ib}^b(k) \end{bmatrix}, \hat{\mathbf{x}}_p = \begin{bmatrix} \mathbf{C}_e^n \mathbf{p}^e(k) \\ \boldsymbol{\Psi}^n(k) \end{bmatrix} = \begin{bmatrix} \mathbf{p}^n(k) \\ \boldsymbol{\Psi}^n(k) \end{bmatrix} \tag{21.66}$$

$$\hat{\mathbf{x}}_{\text{aug}} = \begin{bmatrix} \hat{\mathbf{x}}_v(k) \\ \mathbf{m}^n(k) \\ \hat{\mathbf{x}}_p(k) \end{bmatrix} \tag{21.67}$$

The state covariance matrix is then augmented with the stored vehicle pose:

$$\mathbf{P}_{\text{aug}}(k) = \begin{bmatrix} \mathbf{P}_{vv} & \mathbf{P}_{vm} & \mathbf{P}_{vp} \\ \mathbf{P}_{mv} & \mathbf{P}_{mm} & \mathbf{P}_{mp} \\ \mathbf{P}_{pv} & \mathbf{P}_{pm} & \mathbf{P}_{pp} \end{bmatrix} \tag{21.68}$$

$\hat{\mathbf{x}}_v$ is the concatenation of the vehicle position, velocity, attitude, and inertial sensor bias states where $\hat{\mathbf{x}}_p$ is the concatenation of the vehicle position and attitude (i.e., the vehicle pose states) at the time of the observation. $\hat{\mathbf{x}}_{\text{aug}}$ is the augmented state vector which is comprised of the vehicle states ($\hat{\mathbf{x}}_v$), the 3D positions of all of the map features ($\mathbf{m}^n$), and the added vehicle pose states ($\hat{\mathbf{x}}_p$). The observation (coordinates of the feature in the image plane) is stored separately from the EKF state vector.

In local SLAM, the covariance term $\mathbf{P}_{pp}$ (covariance of the pose states) is derived by taking the position and attitude covariance matrix subblocks from within $\mathbf{P}_{vv}$ (since these states have the same value and the same covariance as the current vehicle position and attitude). Similarly the covariance subblock $\mathbf{P}_{pm}$ is taken from the existing cross correlations between the current vehicle states and map states (i.e., subblocks of $\mathbf{P}_{vm}$ corresponding just to position and attitude). $\mathbf{P}_{pv}$ is the cross correlation between all of the current vehicle states (i.e., position, attitude, velocity,

and inertial sensor biases) and the current pose states. This is thus taken from subblocks of $\mathbf{P}_{vv}$ itself since there are states in common (position and attitude), and thus parts of the added vehicle pose states ($\hat{\mathbf{x}}_p$) are completely correlated to the current vehicle state ($\hat{\mathbf{x}}_v$). In the case of global SLAM, a $\mathbf{C}_e^n$ transformation is applied to the vehicle position covariances in order to represent the stored pose in the local navigation frame.

As the process model of the vehicle comes into play and the filter moves forward in time, the correlations between the stored pose and the new time vehicle states (i.e., $\hat{\mathbf{x}}_v(k+1)$) will decrease (due to the process noise on $\hat{\mathbf{x}}_v(k+1)$).

21.4.2.2 Deciding When to Initialize a Feature

Eventually enough feature observations will be made from varying vehicle poses to initialize the position of the feature. Initializing a feature too early (i.e., by not using enough observations or observations with insufficient separating angle) can result in inconsistency as the true probability distribution of the feature location is not well represented by a Gaussian. The disadvantage with overdelaying the initialization is that the uncertainty in the vehicle states continues to grow before the initialization. As the uncertainty in the current vehicle state grows, linearization errors can affect the consistency of the filter. The information should be initialized and recovered as quickly as possible to limit the effect the inconsistency can have. Additionally, it may be desired to quickly recover the stored information which contributes toward the accuracy of the vehicle localization estimates which may be used as feedback for the control of the vehicle. Another upper limit on deciding how long to delay the initialization is driven by reducing the increased computational burden imposed by adding stored observations to the state vector.

In Bailey (2003), the author discusses a method for testing the conditioning of the initialization by using a Kullback-Leibler distance measure between the linearized update and an approximation of the update using particles to represent the final probability distribution of the feature position. This method however is very computationally intensive; instead, a more practical approach is to set a conservative threshold for the minimum angle between observations necessary to initialize a feature.

21.4.2.3 Initializing a 3D Feature Position Estimate

When it is decided to initialize the 3D position of a feature into the map, the two stored observations of the feature which are separated by the largest angle are used to create an initial estimate of the feature position. Each bearing-only observation can be represented by a 3D point in space $\mathbf{y}^n$ from where the observation was made (at the origin of the sensor) along with a unit vector $\bar{\mathbf{u}}^n$ pointing along the line of sight of the observation; thus,

$$\mathbf{y}^n = \mathbf{p}^n + C_b^n \mathbf{p}_{sb}^b \tag{21.69}$$

$$\bar{\mathbf{u}}^n = C_b^n C_s^b \bar{\mathbf{p}}_{ms}^s \tag{21.70}$$

where $\bar{\mathbf{x}}$ indicates the unit vector of a vector $\mathbf{x}$. $\mathbf{p}^n$ and C_b^n are determined from the stored pose data associated to each observation, and $\bar{\mathbf{p}}_{ms}^s$ is determined from the observation data itself using Eq. 21.44 to convert the pixel observation to azimuth and elevation angles and Eq. 21.71 to convert to a unit vector:

$$\bar{\mathbf{p}}_{ms}^s = \begin{bmatrix} \cos(\varphi_i)\cos(\vartheta_i) \\ \sin(\varphi_i)\cos(\vartheta_i) \\ \sin(\vartheta_i) \end{bmatrix} \tag{21.71}$$

The lines of sight generated by each observation should intersect at one point in 3D space corresponding to the feature location. Since the observations and stored vehicle pose information is noisy, the lines of sight will generally not intersect perfectly. Instead, the initial feature position is computed as the closest point between the two lines for each observation:

$$\begin{aligned} \mathbf{m}_i^n &= \mathbf{G}(\mathbf{p}_1^n, \mathbf{p}_2^n, \boldsymbol{\Psi}_1^n, \boldsymbol{\Psi}_2^n, \mathbf{z}_1, \mathbf{z}_2) \\ &= \frac{1}{2}(\mathbf{y}_1^n + \mathbf{y}_2^n + p_1.\bar{\mathbf{u}}_1^n + p_2.\bar{\mathbf{u}}_2^n) \end{aligned} \tag{21.72}$$

$$p_1 = \frac{((\mathbf{y}_2^n - \mathbf{y}_1^n) \times \bar{\mathbf{u}}_2^n) \cdot (\bar{\mathbf{u}}_1^n \times \bar{\mathbf{u}}_2^n)}{|\bar{\mathbf{u}}_1^n \times \bar{\mathbf{u}}_2^n|^2} \tag{21.73}$$

$$p_2 = \frac{((\mathbf{y}_1^n - \mathbf{y}_2^n) \times \bar{\mathbf{u}}_1^n) \cdot (\bar{\mathbf{u}}_2^n \times \bar{\mathbf{u}}_1^n)}{|\bar{\mathbf{u}}_2^n \times \bar{\mathbf{u}}_1^n|^2} \tag{21.74}$$

In the event that there is a large discrepancy between the two lines (i.e., the minimum distance between the closest two points, one on each line, is larger than a threshold), the observations may be incorrect. This could be due to a misassociation of one of the observations or if the feature is moving for some reason. In this case, the observations are discarded, and the feature is not initialized. Provided there is no large discrepancy between the lines, the state vector and covariance matrix in the SLAM filter are then augmented to include the estimate of the new feature:

$$\hat{\mathbf{x}}_{\text{aug}}(k) = \begin{bmatrix} \hat{\mathbf{x}}(k) \\ \mathbf{m}_i^n(k) \end{bmatrix} \tag{21.75}$$

$$\mathbf{P}_{\text{aug}}(k) = \begin{bmatrix} \mathbf{I} & 0 \\ \nabla\mathbf{G}_p & \nabla\mathbf{G}_z \end{bmatrix} \begin{bmatrix} \mathbf{P}(k) & 0 \\ 0 & \mathbf{R}_{2\times 2} \end{bmatrix} \begin{bmatrix} \mathbf{I} & 0 \\ \nabla\mathbf{G}_p & \nabla\mathbf{G}_z \end{bmatrix}^T \tag{21.76}$$

where $\nabla\mathbf{G}_p$ and $\nabla\mathbf{G}_z$ are the Jacobians of the initialization function $\mathbf{G}(.)$ with respect to the pose states $(\mathbf{p}_1^n, \mathbf{p}_2^n, \boldsymbol{\Psi}_1^n, \boldsymbol{\Psi}_2^n)$ and the observations $(\mathbf{z}_1, \mathbf{z}_2)$, respectively, and $\mathbf{R}_{2\times 2}$ is

$$\mathbf{R}_{2\times 2} = \begin{bmatrix} \mathbf{R} & 0 \\ 0 & \mathbf{R} \end{bmatrix} \tag{21.77}$$

21.4.2.4 Recovering the Information from Remaining Stored Observations

Once two observations have been used to initialize the 3D position of the feature into the filter, the remaining stored observations $(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_j)$ are run through a batch EKF update. The update corrects not only the current feature being initialized but also the other features in the map and the current vehicle state estimates. The updated state vector and state covariance is calculated using EKF update equations described above for the range-bearing algorithm (i.e., Eqs. 21.56 and 21.57) where the innovation $\nu(k)$ is composed using all of the stored observations for the feature:

$$\nu(k) = \begin{bmatrix} \mathbf{z}_{1,\text{pix}} - \mathbf{H}(\mathbf{p}_1^n, \boldsymbol{\Psi}_1^n, \mathbf{m}_i^n) \\ \mathbf{z}_{2,\text{pix}} - \mathbf{H}(\mathbf{p}_2^n, \boldsymbol{\Psi}_2^n, \mathbf{m}_i^n) \\ \vdots \\ \mathbf{z}_{j,\text{pix}} - \mathbf{H}(\mathbf{p}_j^n, \boldsymbol{\Psi}_j^n, \mathbf{m}_i^n) \end{bmatrix} \tag{21.78}$$

Once the update has been completed, pose states that no longer have any associated stored observations been removed from the state vector and their corresponding rows and columns removed from the covariance matrix. Finally, in the case of global SLAM, the newly initialized feature position is transformed from the local navigation frame into the ECEF frame. If local SLAM is performed, the feature position remains in the local navigation frame.

21.4.3 Data Association

When performing data association for features that have already been initialized into the map, the same methods are used as in the range/bearing SLAM case shown in Sect. 21.3.3. Issues arise when attempting to find a data association test that can be performed for uninitialized features. Since the exact 3D position of the feature is not known, one cannot consistently calculate the innovation or innovation covariance of the feature. Instead, from only one or a small number of observations with a small baseline, the observation could lie anywhere in 3D space along the line-of-sight of the observation.

In order to associate observations of features that have not yet been initialized into the 3D map, a multi-hypothesis of Gaussian distributions is created for the possible 3D locations of the feature along the line of sight vector for the first observation of the feature.

21.4.3.1 Starting a New Feature

When an observation is made in the image that cannot be associated to any other previously seen feature, initialized or uninitialized, it is assumed that this observation has come from a new feature that has not been seen before. The process begins by storing the observation and augmenting the EKF state vector with the current vehicle pose (see Sect. 21.4.2.1).

From the single observation, a set of equally weighted hypotheses are created for where the feature could lie in 3D space along the line of sight. The mean ($\hat{\mathbf{x}}_j$) and covariance ($\mathbf{P}_j$) for each hypothesis are calculated for several different values of range (r_j) in equal increments from an expected minimum and maximum sensor range as shown in the left of Fig. 21.4 using Eqs. 21.79 and 21.80:

$$\begin{aligned}\hat{\mathbf{x}}_j &= \mathbf{G}(\mathbf{p}^n(k), \boldsymbol{\Psi}^n(k), \mathbf{z}_i(k), r_j) \\ &= \mathbf{p}^n + C_b^n \mathbf{p}_{sb}^b + r_j.(C_b^n C_s^b \mathbf{p}_{ms}^s) \qquad (21.79)\\ \mathbf{P}_j &= \nabla\mathbf{G}_v \mathbf{P}_{vv} \nabla\mathbf{G}_v^T + \nabla\mathbf{G}_z \mathbf{R}_{\text{ang}} \nabla\mathbf{G}_z^T \qquad (21.80)\end{aligned}$$

where $\mathbf{p}_{ms}^s$ is calculated from the observation data using Eq. 21.71 and $\nabla\mathbf{G}_v$, $\nabla\mathbf{G}_z$ are the Jacobians of the function $\mathbf{G}(.)$ with respect to vehicle states and the observation and range data, respectively. The number of hypotheses used and the maximum and minimum range and thus the spacing between the hypotheses depend on the desired accuracy in the initial feature position with more hypotheses resulting in a better initialization. A record of the multi-hypothesis distribution is maintained separately from the state vector and is used only to assist in associating future observations of the feature.

21.4.3.2 Associating Future Observations and Maintaining Feature Hypotheses

Since each hypothesis is Gaussian with a mean defined in 3D space, the innovation and innovation covariance can be calculated for each hypotheses for each uninitialized feature using

$$\nu(k) = \mathbf{z}_{i,\text{ang}}(k) - \mathbf{H}(\mathbf{p}^n(k), \boldsymbol{\Psi}^n(k), \hat{\mathbf{x}}_j) \qquad (21.81)$$

$$\mathbf{S}(k) = \nabla\mathbf{H}_x(k)\mathbf{P}_{\text{hyp}}(k)\nabla\mathbf{H}_x^T(k) + \mathbf{R}_{\text{ang}} \qquad (21.82)$$

$$\mathbf{P}_{\text{hyp}}(k) = \begin{bmatrix} \mathbf{P}_{pp} & 0 \\ 0 & \mathbf{P}_j \end{bmatrix} \qquad (21.83)$$

where $\mathbf{P}_{pp}$ is the covariance subblock of the current vehicle position and attitude states. An approximation is made that the current vehicle state and the hypothesis are uncorrelated in order to simplify the data association process; to account for these correlations in the data association process would require a large computational resource. The result of this approximation is that during large SLAM loop closures, the EKF may not associate observations of a new feature and some observations may be discarded. It may thus take longer for new features at this time to be integrated into the map. For a given observation and for each hypothesis for a given uninitialized feature, γ is calculated using Eqs. 21.55, 21.81, and 21.82. If the value of γ is below the threshold corresponding to a 95 % confidence for at least one of the hypotheses, then the observation is matched to this uninitialized feature.

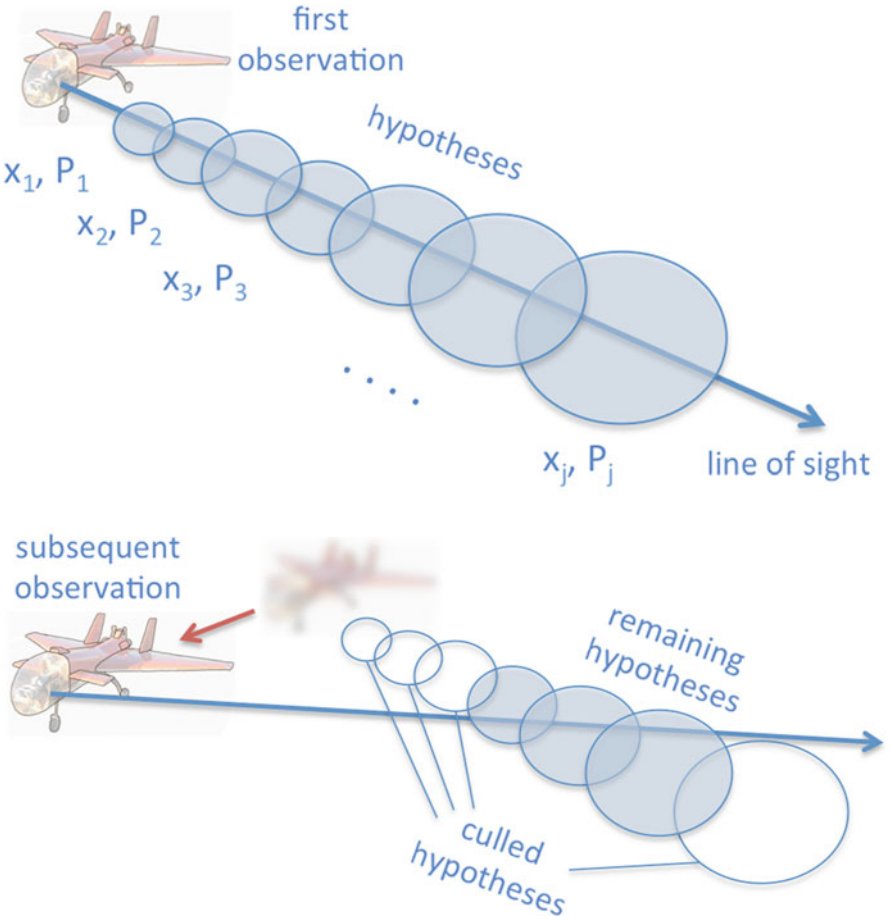

Fig. 21.4 Data association of observations to uninitialized features. When a feature is seen for the first time, a set of hypotheses for the 3D position of the feature is generated at equal range increments along the line of sight (*top*). Future observations are checked for matches to any of the hypotheses. When a match is made to one of the hypotheses, the remaining hypotheses that do not match are culled (*bottom*)

In order to simplify the association of uninitialized features, when an association is made between an observation and one of the hypotheses for a given feature, all other hypotheses of this feature for which the observation does not match are culled from the set of hypotheses from which to associate future observations. As the vehicle moves around an uninitialized feature, the number of hypotheses gradually drops until only one hypothesis matches, the one that is closest to the true 3D feature location. The bottom sub-figure of Fig. 21.4 illustrates this process.

21.4.3.3 Data Association Procedure

Each time observations from the feature extraction process are received, the procedure begins by using Eq. 21.55 to evaluate the potential matching between

each observation and each of the 3D initialized features. Observations that match 3D initialized features are associated and sent on to the SLAM filter to be updated. In the event of multiple features matching a single observation, the matching with the lowest value of γ will be accepted.

The remaining observations are tested for matches with each of the hypotheses for each uninitialized feature. Observations that match with at least one hypothesis of an uninitialized feature are associated to this feature. The observation itself is stored, and the vehicle pose at the current time is then added to the state vector (see Sect. 21.4.2.1). In the event of multiple uninitialized features matching a single observation, all matchings to this observation are rejected.

For each remaining observation not matched to an initialized or un-initialized feature, a new set of hypotheses is created (see Sect. 21.4.3.1).

The proposed multi-hypothesis method for data association could also potentially be used for initializing the feature position, as has been demonstrated in Kwok and Dissanayake (2004), where hypotheses are pruned until only one is left, which then becomes the initialized feature. In this approach, the line-of-sight intersection method is used for calculating the initial feature position; in order to achieve the same accuracy with the multi-hypothesis method, one would require a prohibitively large number of hypotheses (i.e., 1-m resolution for a feature at a range of 200 m would require 200 hypotheses).

21.5 Multi-vehicle Inertial SLAM Algorithm

In multi-vehicle SLAM, several vehicles move over a common section of terrain where the task is to build a common map of the environment while providing localization estimates to each platform. The use of multiple cooperating vehicles has many advantages over SLAM on a single vehicle. Multiple vehicles provide wider sensor coverage and thus can be used to build more extensive terrain maps in less time. The accuracy of the constructed terrain map is greater than in the single-vehicle case as multiple vehicles contribute information toward a given feature location. This increased terrain map accuracy also creates an increase in the accuracy of the localization estimates for each vehicle.

The work by Fenwick et al. (2002), Mourikis and Roumeliotis (2005), Walter and Leonard (2004), and Thrun and Lui (2003a) provides examples of multi-vehicle SLAM where all vehicles send their sensor data to a central Kalman filter. These approaches are fully centralized, involving the communication of raw sensor data from each vehicle to a central source and thus requiring a large amount of communication bandwidth. In Nettleton et al. (2003) and Ong et al. (2003), the authors demonstrate a decentralized architecture for multi-vehicle SLAM using a mathematically equivalent form of the EKF known as the Extended Information Filter (EIF). The use of the EIF in these approaches avoids the need to communicate raw sensor data, where instead this raw data is used in SLAM by each vehicle locally before communicating the map information.

In this section, data fusion architectures are considered for sharing map information built by each vehicle using the inertial SLAM algorithms discussed in Sects. 21.3 and 21.4. The core principle behind the data fusion schemes is the use of the EIF which allows for the map data contributions from each platform to be summed together where the processes of the combined estimation task are distributed among the vehicles. Both centralized and decentralized architectures are discussed in Sects. 21.5.2 and 21.5.3.

21.5.1 Global Vs. Local SLAM for Multiple Vehicles

When global SLAM is performed by each of the vehicles in the data fusion network, each vehicle uses the ECEF frame for referencing the position of terrain features in the environment. In the case of local SLAM, however, each vehicle may use its own independent local navigation frame in which it builds its local map. In order to fuse information from multiple maps, the relative transformations between each local navigation frame must be known. In the case where no localization or prior terrain reference information is available, the vehicles can compute a relative transformation by matching at least two features from each local map (for an example of this process, the reader is referred to Fox et al. (2006)). Once the transformation is known, a common representation of the environment can be built with respect to each vehicle's local navigation frame.

21.5.2 Centralized Architectures for Multi-vehicle Inertial SLAM

This section presents a centralized, distributed data fusion architecture for multi-vehicle inertial SLAM. The architecture is centralized, where some part of the data fusion process is performed at a central node. The architecture is also distributed; rather than communicate all of the raw sensor data from both the inertial and terrain sensors to a central data fusion source, each vehicle firstly performs single-vehicle inertial SLAM as shown in Sects. 21.3 and 21.4. The local terrain map built on each vehicle is then communicated in information form to a central source at regular intervals where data fusion is performed. Finally, the central data fusion node communicates the fused map information back to each vehicle, where this information is fused back into the local map. The following subsections describe the process in more detail.

21.5.2.1 Centralized, Distributed Data Fusion

The centralized, distributed data fusion is based on the independent opinion pool architecture shown in Manyika and Durrant-Whyte (1994). At regular intervals, each vehicle takes its current state estimate relating to the map estimates only, that is, $\mathbf{x}_m(k)$ and $\mathbf{P}_{mm}(k)$, where

$$\mathbf{x}_m(k) = \begin{bmatrix} \mathbf{m}_1^e(k) \\ \mathbf{m}_2^e(k) \\ \vdots \\ \mathbf{m}_N^e(k) \end{bmatrix} \tag{21.84}$$

for the case of global SLAM and

$$\mathbf{x}_m(k) = \begin{bmatrix} \mathbf{m}_1^n(k) \\ \mathbf{m}_2^n(k) \\ \vdots \\ \mathbf{m}_N^n(k) \end{bmatrix} \tag{21.85}$$

for the case of local SLAM. $\mathbf{P}_{mm}(k)$ is a $3N \times 3N$ matrix of the elements of $\mathbf{P}_k$ relating to the map feature estimates. Each vehicle then calculates its posterior information:

$$\mathbf{Y}_j(k) = \mathbf{P}_{mm}^{-1}(k) \tag{21.86}$$

$$\mathbf{y}_j(k) = \mathbf{Y}_j(k)\mathbf{x}_m(k) \tag{21.87}$$

for the jth vehicle where $j = 1, \ldots, M$, where M is the number of vehicles, and communicates this to the central map filter. The information that is sent will obviously be correlated to the information that was sent in the previous communication (since each vehicle's posterior information is based on the entire history of observations it has made). To overcome this, the central data filter maintains a record of the information that it has been sent in the previous communication ($\mathbf{Y}_j(k-1)$,$\mathbf{y}_j(k-1)$) by each vehicle. When the new information arrives, the old information is subtracted from it before adding it to the central map information, in order to remove the correlations and only count new information. The central map information update at the central data filter is thus

$$\mathbf{Y}_{\text{central}}(k) = \mathbf{Y}_{\text{central}}(k-1) + \sum_{j=1}^{M} (\mathbf{Y}_j(k) - \mathbf{Y}_j(k-1)) \tag{21.88}$$

$$\mathbf{y}_{\text{central}}(k) = \mathbf{y}_{\text{central}}(k-1) + \sum_{j=1}^{M} (\mathbf{y}_j(k) - \mathbf{y}_j(k-1)) \tag{21.89}$$

Once the information is combined in the central filter, a state-space estimate of the map feature locations and covariance can be recovered using Eqs. 21.98 and 21.99:

$$\mathbf{P}_{mm,\text{central}}(k) = \mathbf{Y}_{\text{central}}^{-1}(k) \tag{21.90}$$

$$\mathbf{x}_{m,\text{central}}(k) = \mathbf{P}_{mm,\text{central}}(k)\mathbf{y}_{\text{central}}(k) \tag{21.91}$$

21.5.2.2 Applying Local Node Feedback to the Independent Opinion Pool

So that each vehicle's localization estimates can benefit from the observations of features made by other vehicles, information about the central map should be fed back to each of the local nodes. In the same way that was done on the central data filter, each vehicle must store the last information update that it received from the central filter ($\mathbf{Y}_{\text{central}}(k-1)$,$\mathbf{y}_{\text{central}}(k-1)$) so as not to double count the information that has been sent to it. Thus, when each vehicle receives the communicated central information, it firstly computes its posterior information over the entire state space consisting of local vehicle estimate and map features using Eqs. 21.86 and 21.87 and updates this information using Eqs. 21.92 and 21.93:

$$\mathbf{Y}_{\text{local}}(k) = \mathbf{Y}_{\text{local}}(k) + (\mathbf{Y}_{\text{central}}(k) - \mathbf{Y}_{\text{central}}(k-1)) \tag{21.92}$$

$$\mathbf{y}_{\text{local}}(k) = \mathbf{y}_{\text{local}}(k) + (\mathbf{y}_{\text{central}}(k) - \mathbf{y}_{\text{central}}(k-1)) \tag{21.93}$$

The local information is then transformed back into state-space and covariance form to provide the updated estimate of the vehicle localization and map features, which is substituted back into the EKF in the single-vehicle SLAM architecture. The operation of the central filter with local node feedback is illustrated in Fig. 21.5.

This centralized, distributed architecture has several advantages over a completely centralized filter such as a reduction in the required communication bandwidth (as only local estimates must be communicated, not observations and process model inputs) and the ability to deal with intermittent communications and delays as the information is maintained on the local vehicle.

21.5.3 Decentralized Architectures for Multi-vehicle Inertial SLAM

The multi-vehicle inertial SLAM algorithm can be decentralized by removing the central filter, where each vehicle now communicates directly to each other vehicle in the network. This type of architecture was demonstrated for feature tracking tasks and SLAM in Nettleton (2003). At regular intervals each UAV takes its current state estimate relating to the map estimates only, that is, $\mathbf{x}_m(k)$ and $\mathbf{P}_{mm}(k)$, and calculates its posterior information using Eqs. 21.86 and 21.87. Each UAV maintains a record of the information sent during the last communication (i.e., $\mathbf{Y}_j(k-1)$,$\mathbf{y}_j(k-1)$) which is subtracted from the current information to form the new information that UAV has about the feature map:

$$\mathbf{Y}_{j,\text{new}}(k) = \mathbf{Y}_j(k) - \mathbf{Y}_j(k-1) \tag{21.94}$$

$$\mathbf{y}_{j,\text{new}}(k) = \mathbf{y}_j(k) - \mathbf{y}_j(k-1) \tag{21.95}$$

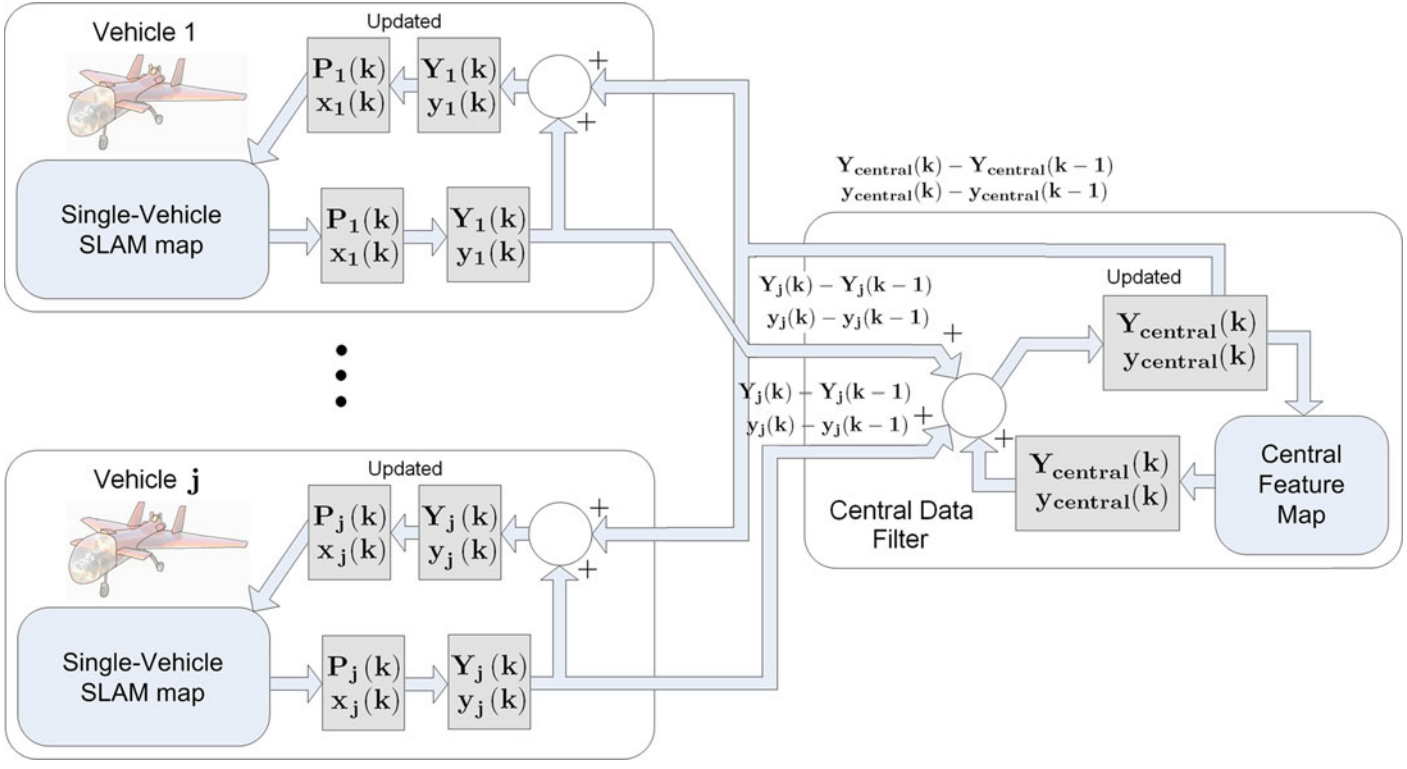

Fig. 21.5 Distributed centralized multi-vehicle SLAM: independent opinion pool architecture with local node feedback. Each vehicle communicates its posterior map estimates in information form which are added together at a central data filter. The central data filter then feeds back the information to each of the vehicles in order to update their local maps

This new information is then communicated to each of the other UAVs. When each UAV receives all of the information updates from each of the other UAVs, this information is summed together along with the current UAV information to form the updated estimate of the map features in information form:

$$\mathbf{Y}_{j,\text{update}}(k) = \mathbf{Y}_j(k) + \sum_{i=1}^{M} \mathbf{Y}_{i,\text{new}}(k) \tag{21.96}$$

$$\mathbf{y}_{j,\text{update}}(k) = \mathbf{y}_j(k) + \sum_{i=1}^{M} \mathbf{y}_{i,\text{new}}(k) \tag{21.97}$$

Once all of the information from other UAVs is combined in the update, a state-space estimate of the map feature locations and covariance can be recovered back into the EKF using Eqs. 21.98 and 21.99:

$$\mathbf{P}_{j,mm,\text{update}}(k) = \mathbf{Y}^{-1}_{j,\text{update}}(k) \tag{21.98}$$

$$\mathbf{x}_{j,m,\text{update}}(k) = \mathbf{P}_{j,mm,\text{update}}(k)\mathbf{y}_{j,\text{update}}(k) \tag{21.99}$$

The operation of the decentralized SLAM filter is illustrated in Fig. 21.6.

21.5.4 Delayed Observations, Network Outages, and Communication Bandwidth Constraints

Realistic communication networks between the vehicles will not be able to provide continuous and instantaneous communication of information. Instead, real networks will contain significant delays and outages between different nodes when vehicles move out of range of one another and will not always be able to communicate all of the map information when the map becomes very large.

Delayed observations are not a significant issue in multi-vehicle SLAM as features are considered stationary, and thus information about a feature's location is independent of time and can be added in a delayed fashion and out of order. When there are outages in the network communications, this can cause the vehicles to lose track of the common information they possess. This can be overcome by constraining the structure of the communications network to tree structures (Nettleton 2003). When communication bandwidth constraints apply across the network, the vehicles may only communicate information representing a subset of the features contained in their map. In this case, information must be fused together using the covariance intersect algorithm (Julier and Uhlmann 2001) due to correlations between the submap and the rest of the map features, which is not accounted for in further communications. These concepts are all discussed further in Nettleton (2003).

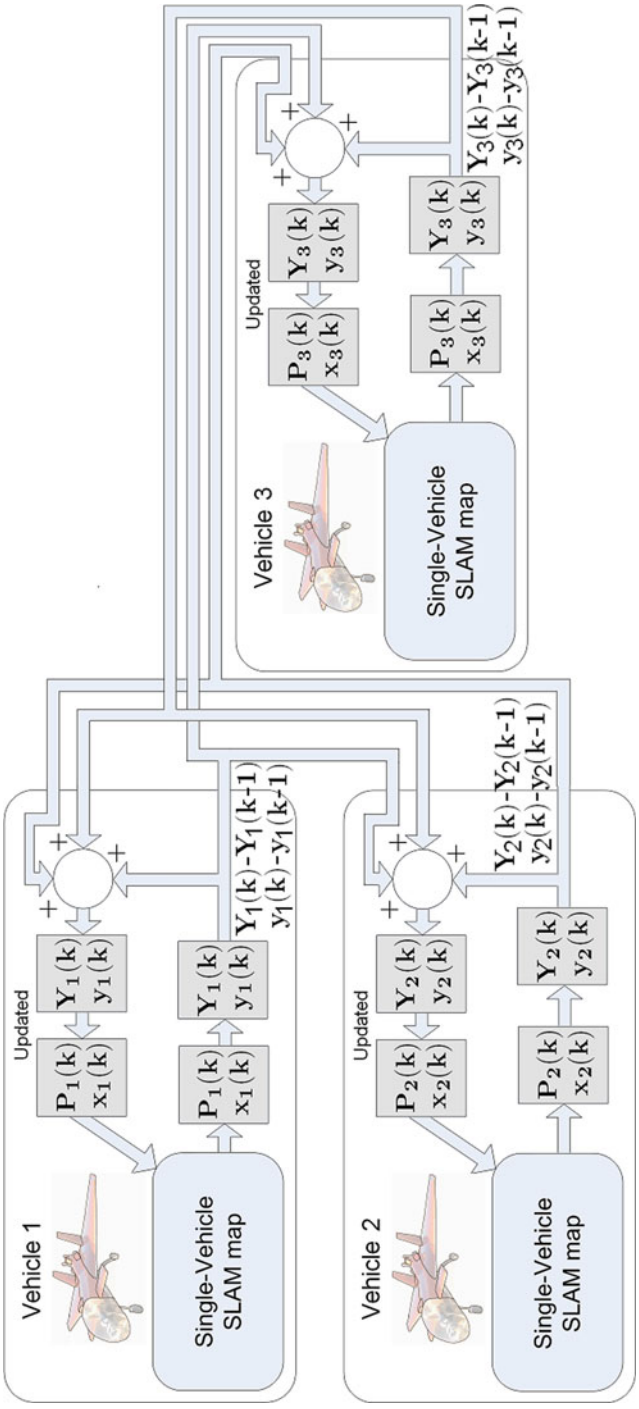

Fig. 21.6 Decentralized SLAM data fusion architecture. Each UAV communicates to each other UAV in the team its latest map state estimate in inverse covariance (information) form. Information is then added at the receiving end and converted from information space back into state space into the EKF

21.6 Conclusion

This chapter has developed the basic equations and methods for inertial sensor-based SLAM. The fundamental equations that model inertial sensors and inertial localization were analyzed; two different applications were examined for when localization and mapping is performed in either a global frame of reference or in an arbitrary local frame of reference. The inertial SLAM algorithms were examined for the case of range and bearing observations from a terrain sensor and also when terrain observations were made from a bearing-only sensor. Finally, the problem of multi-vehicle inertial SLAM was examined. Two different types of data fusion architecture were considered: firstly, centralized architectures in which map information is shared among vehicles via a central communications source and, secondly, decentralized architectures where the vehicles communicate and share data with each other directly.

References

T. Bailey, Constrained initialisation for bearing-only SLAM, in *IEEE International Conference on Robotics and Automation*, Institute of Electrical Engineers, Piscataway, USA (2003)

I. Bar-Itzhack, D. Serfaty, Y. Vitek, Doppler-aided low-accuracy strapdown inertial navigation system. J. Guid. Control **5**(3), 236–242 (1982)

M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, S. Teller, An Atlas framework for scalable mapping, in *IEEE International Conference on Robotics and Automation*, Institute of Electrical Engineers, Piscataway, USA (2003)

M. Bryson, S. Sukkarieh, Building a Robust implementation of bearing-only inertial SLAM for a UAV. J. Field Robot. Spl. Issue SLAM Field **24**(2), 113–143 (2007)

M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, M. Csorba, A solution to the simultaneous localization and map building (SLAM) problem. IEEE Trans. Robot. Autom. **17**(3), 229–241 (2001)

J. Fenwick, P. Newman, J. Leonard, Cooperative concurrent mapping and localisation, in *IEEE International Conference on Robotics and Automation*, Institute of Electrical Engineers, Piscataway, USA (2002)

D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Shculz, B. Stewart, Distributed multirobot exploration and mapping. Proc. IEEE **94**(7), 1325–1339 (2006)

C.S. Giovanni, Performance of a ring laser strapdown attitude and heading reference for aircraft. J. Guid. Control **2**(4), 320–327 (1979)

J. Guivant, E. Nebot, Optimization of the simultaneous localization and map-building algorithm for real-time implementation. IEEE Trans. Robot. Autom. **17**(3), 242–257 (2001)

J. Gutmann, K. Konolige, Incremental mapping of large cyclic environments, in *International Symposium on Computational Intelligence in Robotics and Automation*, Institute of Electrical Engineers, Piscataway, USA (1999)

S. Julier, J. Uhlmann, *General Decentralised Data Fusion with Covariance Intersection (CI): Handbook of Data Fusion* (CRC, Boca Raton, 2001)

J. Kim, S. Sukkarieh, Airborne simultaneous localization and map building, in *IEEE International Conference on Robotics and Automation*, Institute of Electrical Engineers, Piscataway, USA (2003)

N. Kwok, G. Dissanayake, An efficient multiple hypothesis filter for bearing only SLAM, in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Institute of Electrical Engineers, Piscataway, USA (2004)

D. Lowe, Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)

J. Manyika, H. Durrant-Whyte, *Data Fusion and Sensor Management: a decentralised information-theoretic approach*, (Ellis Horwood Ltd., London, 1994)

J. Meyer-Hilberg, T. Jacob, High accuracy navigation and landing system using GPS/IMU system integration, in *IEEE Position, Location and Navigation Symposium*, Institute of Electrical Engineers, Piscataway, USA (1994)

A. Mourikis, S. Roumeliotis, Performance bounds for cooperative simultaneous localisation and mapping (C-SLAM), in *Robotics: Science and Systems Conference*, http://www.roboticsproceedings.org/rss01/p10.html (2005)

J. Neira, J. Tardos, Data association in stochastic mapping using the joint compatibility test. IEEE Trans. Robot. Autom. **17**(6), 890–897 (2001)

E. Nettleton, Decentralised Architectures for Tracking and Navigation with Multiple Flight Vehicles. Ph.d, University of Sydney, 2003

E. Nettleton, S. Thrun, H. Durrant-Whyte, S. Sukkarieh, Decentralised SLAM with low-bandwidth communication for teams of vehicles, in *4th International Conference on Field and Service Robotics*, Springer-Verlag, New York (2003)

M. Nixon, A. Aguado, *Feature Extraction and Image Processing* (Butterworth Heinmann/Newnes, Oxford, 2001)

E. Olson, J. Leonard, S. Teller, Robust range-only Beacon localization, in *IEEE Autonomous Underwater Vehicle Conference*, Institute of Electrical Engineers, Piscataway, USA (2004)

S. Ong, M. Ridley, J. Kim, E. Nettleton, S. Sukkarieh, Six DoF decentralised SLAM, in *Australiasian Conference on Robotics and Automation*, http://www.araa.asn.au/acra/acra2003/papers/31.pdf (2003)

R. Smith, M. Self, P. Cheeseman, *Estimating Uncertain Spatial Relationships in Robotics: Autonomous Robot Vehicles* (Springer, New York, 1990)

S. Thrun, Y. Lui, Multi-Robot SLAM with sparse extended information filters, in *International Symposium on Robotics Research*, Springer-Verlag, New York (2003a)

S. Thrun, Y. Lui, Results for outdoor-SLAM using sparse extended information filters, in *IEEE International Conference on Robotics and Automation*, Institute of Electrical Engineers, Piscataway, USA (2003b)

D. Titterton, J. Weston, *Strapdown Inertial Navigation Technology* (Peter Peregrinus Ltd., London, 1997)

M. Walter, J. Leonard, An experimental investigation of cooperative SLAM, in *5th International Symposium on Intelligent Autonomous Vehicles* (2004)

S. Williams, G. Dissanayake, H. Durrant-Whyte, Towards terrain-aided navigation for underwater Robotics. Adv. Robot. **15**(5), 533–550 (2001)