Tightly-coupled Visual-Inertial Sensor Fusion based on IMU Pre-Integration

September 11, 2016

Chapter 1

Visual-Inertial Navigation System

1.1 Introduction and Related Work

There have been increasing demands for developing high maneuverability robots, such as micro aerial vehicles (MAVs) with vision-based autonomy for various missions in confined environments. Such robots and sensor suites should be miniaturized, rapidly deployable, and require minimum maintenance even in hazardous environments. The monocular visual-inertial system, which consists of only a low-cost MEMS IMU and a camera, has become a very attractive sensor choice due to its superior size, weight, and power (SWaP) characteristics. In fact, the monocular VINS is the minimum sensor suite that allows both accurate state estimation and sufficient environment awareness.

There is a rich body of scholarly work on VINS state estimation with either monocular [1, 2, 3], stereo [4], or RGB-D cameras [5]. We can categorize solutions to VINS as filtering-based [6, 7, 8, 1, 2, 3],or graph optimization-/bundle adjustment-based [4, 9, 10]. Filtering-based approaches may require fewer computational resources due to the marginalization of past states, but the early fix of linearization points may lead to sub-optimal results. On the other hand, graph optimization-based approaches may improve performance via iterative re-linearization at the expense of higher computational demands. In real-world applications, marginalization [11] is usually employed for both filtering- and optimization-based approaches to achieve constant computational complexity. Conditioning is also a popular method within the computation vision community to enforce constant computation [12]. A comparison between filtering- and graph optimization-based approaches [13] demonstrates nearly identical results. However, the platform for verification is only equipped with an optical flow sensor that is insufficient for extended feature tracking. This limits the power of graph-based approaches, as a well-connected graph is never constructed.

Another way to categorize VINS solutions is to consider them as loosely-coupled [14, 3] or tightly-coupled [6, 7, 8, 1, 2, 4, 9, 15]. Loosely-coupled approaches usually consist of a standalone vision-only state estimation module such as PTAM [12] or SVO [16], and a separate IMU fusion module [14]. These approaches do not consider the visual and inertial information coupling, making them incapable of correcting drifts in the vision-only estimator. Tightly coupled approaches perform systematic fusion of visual and IMU measurements and usually lead to better results [4, 10]. In particular, for the monocular VINS, tightly-coupled methods are able to implicitly incorporate the metric scale information from IMU into scene depth estimation, thus removing the need for explicit visual scale modeling.

In this report, we present the methodology of tightly-coupled approach.

1.2 Sensor Models

Before presenting sensor models, we define notations.

Generally we consider $(\mathbf{o})_Y^X$ as the property $\mathbf{o}$ of frame Y with respect to frame X. The property $\mathbf{o}$ can be position $\mathbf{p}$, velocity $\mathbf{v}$, and orientation $\mathbf{q}$. The frame can be the earth's inertial frame w, the IMU frame b, and the camera frame c. We further denote b_i as the IMU frame while the camera is taking the i^{th} image. As an instance, $\mathbf{p}_{b_i}^w$ is the position

of the IMU frame in world frame when the i^{th} image is captured. The camera-IMU transformation is an unknown constant that we denote as $\mathbf{p}_c^b$ and $\mathbf{q}_c^b$.

Note that IMU usually runs at a much higher rate than the camera, and that multiple IMU measurements may exist in the interval $[i, i+1]$. Specially, $\mathbf{o}_t^X$ represents the property $\mathbf{o}$ of the IMU frame at time t with respect to frame X.

We mainly use quaternions ($\mathbf{q} = [q_w, q_x, q_y, q_z]^\intercal$) in Hamilton notation to represent rotation. $\mathbf{g}^w = [0,\ 0,\ g]^\intercal$ is the gravity vector in the world frame.

Additionally, we use $\tilde{(\cdot)}$ for noisy measurements and $\hat{(\cdot)}$ for the latest estimated value. We also use $\cdot$ for function chaining such that $\mathrm{f} \cdot \mathrm{g} \cdot \mathrm{h}(\mathbf{x}) = (\mathrm{f}(\mathrm{g}(\mathrm{h}(\mathbf{x}))))$

1.2.1 Camera Model

Camera model describes the procedure where a 3D point $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$ is projected to image at $\begin{bmatrix} u \\ v \end{bmatrix}$, where extrinsic and intrinsic parameters play an essential.

The intrinsic parameters establish the relationship of the 2D position between world frame (unit in meter) with image frame (unit in pixel). Usually a pre-process has been done to remove the effect of intrinsic parameters so that the unit can be unified (to be meter) in the following operations.

We mainly focus on how extrinsic parameters are involved in the following camera model:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \mathrm{Proj} \cdot \mathrm{f}_w^c(\begin{bmatrix} x \\ y \\ z \end{bmatrix}), \tag{1.1}$$

where the function $\mathrm{f}_n^m(.)$ that transforms a 3D point $\mathbf{r}$ from frame n to frame m:

$$\mathrm{f}_n^m(\mathbf{r}) = \mathbf{R}_n^m \cdot \mathbf{r} + \mathbf{p}_n^m \tag{1.2}$$

and the function $\mathrm{Proj}(\cdot)$ executes the projection operation:

$$\mathrm{Proj}(\begin{bmatrix} x \\ y \\ z \end{bmatrix}) = \begin{bmatrix} x/z \\ y/z \end{bmatrix} \tag{1.3}$$

It is also necessary to define their inverse functions:

$$\mathrm{f}_m^n(\mathbf{r}) = \mathrm{f}_n^{m^{-1}}(\mathbf{r}) = \mathbf{R}_m^n \cdot (\mathbf{r} - \mathbf{p}_n^m) \tag{1.4}$$

$$\text{Back-Proj}(\begin{bmatrix} u \\ v \end{bmatrix}, \lambda) = \begin{bmatrix} u/\lambda \\ v/\lambda \\ 1/\lambda \end{bmatrix}. \tag{1.5}$$

1.2.2 IMU Model

The measured inertial quantities, linear acceleration and angular velocity $\begin{bmatrix} \tilde{\boldsymbol{\omega}}(t) \\ \tilde{\mathbf{a}}(t) \end{bmatrix}$, are modeled as sum of the true value $\begin{bmatrix} \boldsymbol{\omega}(t) \\ \mathbf{a}(t) \end{bmatrix}$, time-variant bias $\begin{bmatrix} \boldsymbol{\omega}_b(t) \\ \mathbf{a}_b(t) \end{bmatrix}$, and Gaussian white noise $\begin{bmatrix} \boldsymbol{\omega}_n(t) \\ \mathbf{a}_n(t) \end{bmatrix}$:

$$\begin{bmatrix} \tilde{\boldsymbol{\omega}}(t) \\ \tilde{\mathbf{a}}(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\omega}(t) \\ \mathbf{a}(t) \end{bmatrix} + \begin{bmatrix} \boldsymbol{\omega}_b(t) \\ \mathbf{a}_b(t) \end{bmatrix} + \begin{bmatrix} \boldsymbol{\omega}_n(t) \\ \mathbf{a}_n(t) \end{bmatrix} \tag{1.6}$$

Table 1.1: IMU characteristics

Gyroscope			Accelerometer		
$\sigma_{\boldsymbol{\omega}_n}$	1.2×10^{-3}	rad/s$\sqrt{\text{Hz}}$	$\sigma_{\mathbf{a}_n}$	8.0×10^{-3}	m/s$^2\sqrt{\text{Hz}}$
$\sigma_{\boldsymbol{\omega}_r}$	2.0×10^{-5}	rad/s$^2\sqrt{\text{Hz}}$	$\sigma_{\mathbf{a}_r}$	5.5×10^{-5}	m/s$^3\sqrt{\text{Hz}}$

The Gaussian white noise $\begin{bmatrix}\boldsymbol{\omega}_n(t)\\ \mathbf{a}_n(t)\end{bmatrix}$ can be characterized using first and second moments:

$$\begin{aligned} E[\boldsymbol{\omega}_n(t)] &= \mathbf{0}_{3\times 1} \\ E[\boldsymbol{\omega}_n(t+\tau)\boldsymbol{\omega}_n^\intercal(t)] &= \sigma_{\boldsymbol{\omega}_n}\mathbf{I}_{3\times 3}\delta(\tau) \\ E[\mathbf{a}_n(t)] &= \mathbf{0}_{3\times 1} \\ E[\mathbf{a}_n(t+\tau)\mathbf{a}_n^\intercal(t)] &= \sigma_{\mathbf{a}_n}\mathbf{I}_{3\times 3}\delta(\tau), \end{aligned} \tag{1.7}$$

where δ is Dirac function.
The slowly drafting biases vary over time and are modeled as Brownian motions:

$$\begin{bmatrix}\dot{\boldsymbol{\omega}}_n(t)\\ \dot{\mathbf{a}}_n(t)\end{bmatrix} = \begin{bmatrix}\boldsymbol{\omega}_r(t)\\ \mathbf{a}_r(t)\end{bmatrix} \tag{1.8}$$

, where the drafting is driven by Gaussian white noise $\begin{bmatrix}\boldsymbol{\omega}_r(t)\\ \mathbf{a}_r(t)\end{bmatrix}$:

$$\begin{aligned} E[\boldsymbol{\omega}_r(t)] &= \mathbf{0}_{3\times 1} \\ E[\boldsymbol{\omega}_r(t+\tau)\boldsymbol{\omega}_r^\intercal(t)] &= \sigma_{\boldsymbol{\omega}_r}\mathbf{I}_{3\times 3}\delta(\tau) \\ E[\mathbf{a}_r(t)] &= \mathbf{0}_{3\times 1} \\ E[\mathbf{a}_r(t+\tau)\mathbf{a}_r^\intercal(t)] &= \sigma_{\mathbf{a}_r}\mathbf{I}_{3\times 3}\delta(\tau) \end{aligned} \tag{1.9}$$

The noise characteristics for a typical IMU are listed in Table 1.1. In our implementation, discretization is done over the continuous model and will be detailed in Sect. 1.4.2.

1.3 Probabilistic Formulation

In this section, we present our framework in probabilistic language.

1.3.1 Factor Graph

There are many popular probabilistic graphical models in history: Bayesian Network, Hidden Markov Field, Factor Graph, and so on. The tool we choose for our system is factor graph, where nodes and edges are key components for understanding and will be clarified in the following.

Nodes

Nodes in factor graph correspond to random variables. Specifically, in visual-inertial navigation system the variables consist of

1. navigational parameters: position $\mathbf{p}$, velocity $\mathbf{v}$, orientation $\mathbf{q}$, IMU bias $\mathbf{a}_b$ and $\boldsymbol{\omega}_b$.
2. visual parameters: feature inverse depth λ.
3. sensor-sensor parameter: the camera-IMU transformation $\mathbf{p}_c^b$ and $\mathbf{q}_c^b$.

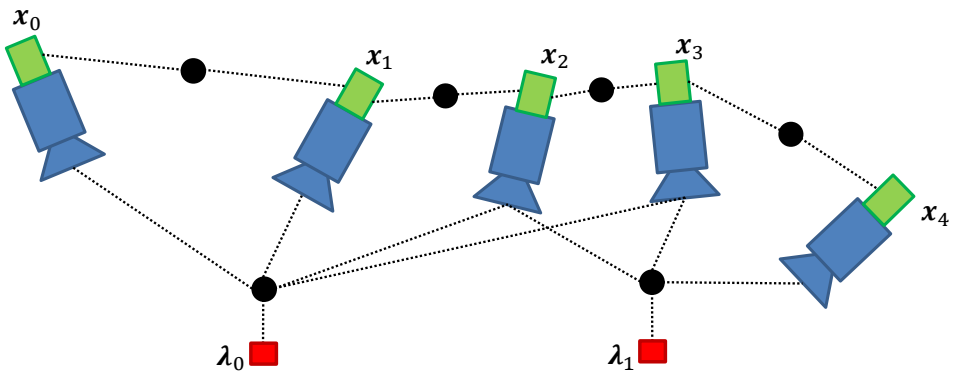


Figure 1.1: A factor graph with five IMU states $\mathbf{x}_k$ and two features $\mathbf{f}_l$. There are two kinds of factors (black circles):Visual factors that connect IMU states and features, IMU factors that connect successive IMU state.

The navigational parameters are naturally sampled at different time along image capture. We include a fixed number of parameters in the factor graph, meaning that the graph is a sliding window containing the latest parameters.

The definition of the full state for a sliding window with N IMU frames and M features is formally listed (the transpose is again ignored):

$$\mathcal{X} = \left[\mathbf{x}_n,\, \mathbf{x}_{n+1},\, \cdots \mathbf{x}_{n+N-1},\, \mathbf{x}_c^b,\, \lambda_m\, \lambda_{m+1}\, \cdots\, \lambda_{m+M-1}\right] \tag{1.10}$$

$$\mathbf{x}_k = \left[\mathbf{p}_{b_k}^w,\, \mathbf{v}_{b_k}^w,\, \mathbf{q}_{b_k}^w, \mathbf{a}_{b_k}, \boldsymbol{\omega}_{b_k}\right] \tag{1.11}$$

$$\mathbf{x}_c^b = \left[\mathbf{p}_c^b,\, \mathbf{q}_c^b\right]. \tag{1.12}$$

Edges

Each conditional distribution abstracts one measurement function in the system. The distribution appears in the graphical model in the form of a groups of edges centered by a factor node.

As illustrated in Fig. (1.1), VINS consists of two kinds of factors. Corresponding conditional distribution of these two kinds of factors are given here for brief introduction of the structure, and detailed derivation will be described in Sect. 1.4.

Each visual factor connects poses of two IMU frames, feature inverse depth, and extrinsic camera-IMU calibration together, forming the following conditional distribution:

$$P(\mathbf{z}_l^{c_j} \mid \mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_c^b, \lambda_l), \tag{1.13}$$

where the vector $\mathbf{z}_l^{c_j}$ results from the procedure where the l^{th} feature is projected to camera frame c_j.

The IMU factor builds the relationship between successive two IMU frame:

$$P(\mathbf{z}_{bi+1}^{b_i} \mid \mathbf{x}_i, \mathbf{x}_{i+1}), \tag{1.14}$$

where the value $\mathbf{z}_{bi+1}^{b_i}$ results from the procedure where the i^{th} IMU frame evolves to the $i+1^{th}$ IMU frame.

1.3.2 Maximum Likelihood Estimation via Nonlinear Optimization

The factor graph directly gives the factorization of the joint distribution:

$$P(\mathbf{z} \mid \mathcal{X}) = \prod_{(l,j)\in\mathcal{C}} P(\mathbf{z}_l^{c_j} \mid \mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_c^b, \lambda_l) \times \prod_{k\in\mathcal{B}} P(\mathbf{z}_{bi+1}^{b_i} \mid \mathbf{x}_i, \mathbf{x}_{i+1}) \tag{1.15}$$

where $\mathcal{C}$ is the set of all observations formed by feature projections and $\mathcal{B}$ is the set of all relative IMU measurements between successive IMU frames.

We assume both distributions to be Gaussian:

$$\begin{aligned} P(\mathbf{z}_l^{c_j} \mid \mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_c^b, \lambda_l) &\sim \mathcal{N}(\tilde{\mathbf{z}}_l^{c_j}, \Sigma_l^{c_j}) \\ P(\mathbf{z}_l^{c_j} \mid \mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_c^b, \lambda_l) &\propto \exp -\frac{1}{2} \left\| \mathbf{z}_l^{c_j} - \tilde{\mathbf{z}}_l^{c_j} \right\|_{\Sigma_l^{c_j}}^2 \\ P(\mathbf{z}_{bi+1}^{b_i} \mid \mathbf{x}_i, \mathbf{x}_{i+1}) &\sim \mathcal{N}(\tilde{\mathbf{z}}_l^{c_j}, \Sigma_{bi+1}^{b_i}) \\ P(\mathbf{z}_{bi+1}^{b_i} \mid \mathbf{x}_i, \mathbf{x}_{i+1}) &\propto \exp -\frac{1}{2} \left\| \mathbf{z}_{bi+1}^{b_i} - \tilde{\mathbf{z}}_{bi+1}^{b_i} \right\|_{\Sigma_{bi+1}^{b_i}}^2 , \end{aligned} \tag{1.16}$$

where the probability of measurement error $\mathbf{r} = \mathbf{z} - \tilde{\mathbf{z}}$ between predicted value $\mathbf{z}$ and noisy measured value $\tilde{\mathbf{z}}$ is proportional to its Mahalanobis norm $\|\mathbf{r}\|_\Sigma^2 = \mathbf{r}^\intercal \Sigma^{-1} \mathbf{r}$.

Then the log likelihood function is in quadratic form:

$$\begin{aligned} \log \mathcal{L}(\mathcal{X} \mid \mathbf{z}) &= \log P(\mathbf{z} \mid \mathcal{X}) \\ &= \sum_{(l,j) \in \mathcal{C}} -\frac{1}{2} \left\| \mathbf{z}_l^{c_j} - \tilde{\mathbf{z}}_l^{c_j} \right\|_{\Sigma_l^{c_j}}^2 + \sum_{k \in \mathcal{B}} -\frac{1}{2} \left\| \mathbf{z}_{bi+1}^{b_i} - \tilde{\mathbf{z}}_{bi+1}^{b_i} \right\|_{\Sigma_{bi+1}^{b_i}}^2 \end{aligned} \tag{1.17}$$

After stacking the residual, we are actually maximizing the L_2 norm of $r(\mathcal{X}) = \mathbf{z} - \tilde{\mathbf{z}}$. Then the maximum likelihood estimation is equivalent to non-linear least squares optimization. We adopt second-order Gauss-Newton method to solve the problem.

Taking the nonlinear function $r(\mathcal{X})$ as an example of nonlinear function of the full state $\mathcal{X}$, we introduce Gauss-Newton method briefly here.

First order Taylor expansion around the latest estimation $\hat{\mathcal{X}}$ gives:

$$r(\mathcal{X}) = r(\tilde{\mathcal{X}} + \delta\mathcal{X}) \approx r(\hat{\mathcal{X}}) + J\delta\mathcal{X} \tag{1.18}$$

where J denotes the Jacobian of the nonlinear function $r_(\mathcal{X})$ with respect to the residual $\delta\mathcal{X}$ around the latest estimation $\tilde{\mathcal{X}}$. Then (1.17), a weighted least square function, can be maximized by iteratively:

1. solving the following linear system,

$$\begin{aligned} \mathbf{A}\delta\mathcal{X} &= \mathbf{b} \\ J^\intercal \Sigma^{-1} J \delta\mathcal{X} &= -J^\intercal \Sigma^{-1} r_{\mathcal{A}}(\hat{\mathcal{X}}) \end{aligned} \tag{1.19}$$

2. updating the latest estimation:

$$\hat{\mathcal{X}} \leftarrow \hat{\mathcal{X}} + \delta\mathcal{X}. \tag{1.20}$$

The equation in (1.19), which is commonly called normal equation, approximates the nonlinear system in a linear form. The linear form not only makes the efficient second-order optimization achievable, but also enables another important operation on the probabilistic formulation: marginalization.

Marginalization of the system will be detailed in Sect. 1.5. Intuitively, marginalization summarizes information in history into a prior distribution. The prior distribution can be integrated into the formulation above naturally, where the maximum likelihood estimation switches to be maximum a-posterior estimation.

1.3.3 More about Optimization

An optimization problem is formulated in Sect. 1.3.2 and Gauss-Newton is suggested to be used. In this part, we further discuss about useful techniques, without which the algorithm can not work in practice.

Error-state Parameterization

The Gauss-Newton mentioned above is design for system where the variable lives in linear space, which is not the case in VINS because of the orientation.

We use quaternion to parameterize orientation using four elements. The over-parameterization makes the normal equation in (1.19) not a positive definite matrix. The so-called error-state parameterization is introduced here to handle the trouble.

The error-state parameterization is inspired by generalized Taylor expansion (1.18). Instead of caring about the original state $\mathcal{X}$, we build the system upon the residual $\delta\mathcal{X}$ around the latest estimation $\hat{\mathcal{X}}$. The only modification we need to do is to replace the addition with quaternion multiplication for rotation:

$$\mathbf{q} = \hat{\mathbf{q}} \times \delta\mathbf{q} = \hat{\mathbf{q}} \times \begin{bmatrix} 1 \\ \frac{1}{2}\delta\boldsymbol{\theta} \end{bmatrix}, \tag{1.21}$$

where $\delta\mathbf{q} = \begin{bmatrix} 1 \\ \frac{1}{2}\delta\boldsymbol{\theta} \end{bmatrix}$ approximates small rotation.

The error-state vector $\delta\mathcal{X}$ is actually what we optimize during the nonlinear optimization:

$$\delta\mathcal{X} = \begin{bmatrix} \delta\mathbf{x}_n,\ \delta\mathbf{x}_{n+1},\ \cdots\ \delta\mathbf{x}_{n+N-1},\ \delta\mathbf{x}_c^b,\ \delta\lambda_m\, \delta\lambda_{m+1}\ \cdots\ \delta\lambda_{m+M-1} \end{bmatrix} \tag{1.22}$$

$$\delta\mathbf{x}_k = \begin{bmatrix} \delta\mathbf{p}_{b_k}^w,\ \delta\mathbf{v}_{b_k}^w,\ \delta\boldsymbol{\theta}_{b_k}^w, \delta\mathbf{a}_{b_k}, \delta\boldsymbol{\omega}_{b_k} \end{bmatrix} \tag{1.23}$$

$$\delta\mathbf{x}_c^b = \begin{bmatrix} \delta\mathbf{p}_c^b,\ \delta\boldsymbol{\theta}_c^b \end{bmatrix}. \tag{1.24}$$

The error-state parameterization also helps model the uncertainty of the orientation, which will be detailed in Sect. 1.4.

Robust Norm

The Guassian assumption is accurate is for IMU factors but is not for Visual factors. Feature tracker suffers outliers commonly because of image noise and ambiguity in screen. L_2 norm used in least square problem is sensitive to outliers. Very small number of outliers will drive the problem away. Thus robust norm is proposed to handle this drawback. Instead of optimizing the L_2 norm of the residual (1.17):

$$\max_{\mathcal{X}} \sum_{(l,j)\in\mathcal{C}} -\frac{1}{2}\left\|\mathbf{z}_l^{c_j} - \tilde{\mathbf{z}}_l^{c_j}\right\|_{\Sigma_l^{c_j}}^2 + \sum_{k\in\mathcal{B}} -\frac{1}{2}\left\|\mathbf{z}_{bi+1}^{b_i} - \tilde{\mathbf{z}}_{bi+1}^{b_i}\right\|_{\Sigma_{bi+1}^{b_i}}^2 \tag{1.25}$$

we apply huber norm [17] to visual factors:

$$\max_{\mathcal{X}} \sum_{(l,j)\in\mathcal{C}} -\frac{1}{2}\left\|\mathbf{z}_l^{c_j} - \tilde{\mathbf{z}}_l^{c_j}\right\|_{\Sigma_l^{c_j}}^2 + \sum_{k\in\mathcal{B}} -\frac{1}{2}\rho\left(\left\|\mathbf{z}_{bi+1}^{b_i} - \tilde{\mathbf{z}}_{bi+1}^{b_i}\right\|_{\Sigma_{bi+1}^{b_i}}^2\right), \tag{1.26}$$

where the huber norm ρ is defined as:

$$\rho(s) = \begin{cases} 1 & s \geq 1, \\ 2\sqrt{s} - 1 & s < 1. \end{cases} \tag{1.27}$$

We used 1 for the threshold because the noise has already been whiten by the covariance matrix Σ during evaluate the Mahalanobis norm.

Damped Method

The numerical stability and efficiency of solving the normal equation (1.19) rely heavily on *how well* the matrix $\mathbf{A}$ is positive definite. In VINS, two facts break the property:

1. Features without sufficient visual parallax can not be safely triangulated. It is unobservable in such a case and results that the matrix $\mathbf{A}$ is semi-definite positive. While theses unobservable features may be harmful to the system, throwing them out is not a good idea because even without accurate depth estimation the rotational constrains from them are still available.

2. The study on observability of visual-inertial fusion [8, 7] clearly points out the initial position and yaw angle are not observable, showing that the dimension of the null space of the matrix $\mathbf{A}$ must be at least 4.

Damped method gives an elegant way to deal with it by solving:

$$(\mathbf{A} + s\mathbf{I})\mathcal{X} = \mathbf{b}, \tag{1.28}$$

where s is a dynamic step size. Intuitively, the additional identity matrix avoids poor condition number, and tends to keep the unobservable variables to be unchanged. Different strategies were proposed to adjust the step size. A comparison between two popular methods Levenberg-Marquardt and Powell's dog leg is given by Manolis et al. [18].

We found that the damped method significantly increases the numerical stability without losing efficiency. Note that the damped method still can not guarantee we reach the optimal solution, and even can not guarantee the cost monotonically decreases iteration by iteration. We return the best result we have ever reached, which corresponds to solution with highest value evaluated by (1.26).

1.4 Factors

Factors used in Factor graph are the most essential parts. For each kinds of factors, detailed description can be done by answering four questions:

1. Which parts of variables are involved? This question declares which parts of nodes in the graph are took into consideration.

2. What is the measurement function? This question abstracts the physical constrains in the system formally.

3. How to formulate the measurement error (usually called as residual) using probabilistic distribution? This question fundamentally enables the probabilistic formulation.

4. How to minimize the likelihood function, given the probabilistic distribution? This question makes use of the probabilistic formulation defined above to achieve the final accurate estimation. As derived in the nonlinear optimization framework Sect. 1.3.2, it is equivalent to Jacobian evaluation of the function with respect to involved variables.

In the following sections, visual and inertial factors are documented by answering the four questions. What is worth to emphasize is that, various kinds of measurements can also be integrated into this system once corresponding factors are formulated, such as GPS, magnetometer and so on.

1.4.1 Visual Factor

Question 1

Each visual measurement is a procedure where the l-th feature, which is firstly observed in camera frame c_i at $\begin{bmatrix} u_l^{c_i} \\ v_l^{c_i} \end{bmatrix}$, is observed again in camera frame c_j at $\begin{bmatrix} \tilde{u}_l^{c_j} \\ \tilde{v}_l^{c_j} \end{bmatrix}$.

Four parts of the state vector $\mathcal{X}$ are involved: inverse depth of the l^{th} feature λ_l, poses of two IMU frames $\begin{bmatrix} \mathbf{p}_{b_i}^w \\ \mathbf{q}_{b_i}^w \end{bmatrix}$, $\begin{bmatrix} \mathbf{p}_{b_j}^w \\ \mathbf{q}_{b_j}^w \end{bmatrix}$, and camera-IMU extrinsic parameter $\begin{bmatrix} \mathbf{p}_c^b \\ \mathbf{q}_c^b \end{bmatrix}$.

Question 2

With the help of the projection function **Proj**, back-projection function **Back-Proj** and frame transform function $\mathbf{f}_n^m$ in Sect 1.2.1, the measurement function is:

$$\begin{bmatrix} u_l^{c_j} \\ v_l^{c_j} \end{bmatrix} = \text{Proj} \cdot \text{f}_c^{b^{-1}} \cdot \text{f}_{b_j}^{b_0^{-1}} \cdot \text{f}_{b_i}^{b_0} \cdot \text{f}_c^b \cdot \text{Back-Proj} \left(\begin{bmatrix} u_l^{c_i} \\ v_l^{c_i} \end{bmatrix}, \lambda_l \right). \tag{1.29}$$

Question 3

The probabilistic distribution corresponding to visual measurements is assumed to be Gaussian:

$$\begin{aligned} r_{\mathcal{C}} \left(\mathbf{x}_{b_i}^w, \mathbf{x}_{b_j}^w, \mathbf{x}_c^b, \lambda_l \right) &= \begin{bmatrix} u_l^{c_j} \\ v_l^{c_j} \end{bmatrix} - \begin{bmatrix} \hat{u}_l^{c_j} \\ \hat{v}_l^{c_j} \end{bmatrix} \\ r_{\mathcal{C}} \left(\mathbf{x}_{b_i}^w, \mathbf{x}_{b_j}^w, \mathbf{x}_c^b, \lambda_l \right) &\sim \mathcal{N}(\mathbf{0}, \Sigma_l^{c_j}) \end{aligned} \tag{1.30}$$

Question 4

The Jacobian is:

$$J_l^{c_j} = \begin{bmatrix} \frac{\partial r_{\mathcal{C}}(\cdot)}{\partial \begin{bmatrix} \delta \mathbf{p}_{b_i}^w \\ \delta \boldsymbol{\theta}_{b_i}^w \end{bmatrix}} & \frac{\partial r_{\mathcal{C}}(\cdot)}{\partial \begin{bmatrix} \delta \mathbf{p}_{b_j}^w \\ \delta \boldsymbol{\theta}_{b_j}^w \end{bmatrix}} & \frac{\partial r_{\mathcal{C}}(\cdot)}{\partial \begin{bmatrix} \delta \mathbf{p}_c^b \\ \delta \boldsymbol{\theta}_c^b \end{bmatrix}} & \frac{\partial r_{\mathcal{C}}(\cdot)}{\partial \delta \lambda_l} \end{bmatrix} \tag{1.31}$$

1.4.2 IMU Pre-Integration Factor

Question 1

Inertial factor connects two successive IMU frames b_i and b_j. The navigational variables $\mathbf{x}_i = \begin{bmatrix} \mathbf{p}_{b_i}^w \\ \mathbf{v}_{b_i}^w \\ \mathbf{q}_{b_i}^w \\ \mathbf{a}_{b_i} \\ \boldsymbol{\omega}_{b_i} \end{bmatrix}$ and $\mathbf{x}_{i+1} = \begin{bmatrix} \mathbf{p}_{b_{i+1}}^w \\ \mathbf{v}_{b_{i+1}}^w \\ \mathbf{q}_{b_{i+1}}^w \\ \mathbf{a}_{b_{i+1}} \\ \boldsymbol{\omega}_{b_{i+1}} \end{bmatrix}$ are involved.

Question 2

Following Newton's Law (we use $\mathbf{a}_t$ for $\mathbf{a}(t)$, and $\boldsymbol{\omega}_t$ for $\boldsymbol{\omega}(t)$ for simplicity):

$$\begin{aligned} \dot{\mathbf{p}}_t^w &= \mathbf{v}_t^w \\ \dot{\mathbf{v}}_t^w &= \mathbf{a}_t^w \\ \dot{\mathbf{q}}_t^w &= \mathbf{q}_t^w \times \begin{bmatrix} 0 \\ \frac{1}{2}\boldsymbol{\omega}_t \end{bmatrix}, \end{aligned} \tag{1.32}$$

the navigational variables $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ in IMU frame b_i and b_{i+1} are constrained by $n+1$ inertial measurements during time interval $[i, i+1]$:

$$\mathbf{p}^w_{b_{i+1}} = \mathbf{p}^w_{b_i} + \mathbf{v}^w_{b_i}\Delta t + \iint_{t\in[i,i+1]} \left(\mathbf{q}^w_t \mathbf{a}_t - \mathbf{g}^w\right) dt^2 \tag{1.33}$$

$$\mathbf{v}^w_{b_{i+1}} = \mathbf{v}^w_{b_i} + \int_{t\in[i,i+1]} \left(\mathbf{q}^w_t \mathbf{a}_t - \mathbf{g}^w\right) dt \tag{1.34}$$

$$\mathbf{q}^w_{b_{i+1}} = \int_{t\in[i,i+1]} \mathbf{q}^w_t \times \begin{bmatrix} 0 \\ \frac{1}{2}\boldsymbol{\omega}_t \end{bmatrix} dt \tag{1.35}$$

where Δt is the duration of time interval $[i, i+1]$.

It can be seen that the rotation between the world frame and the body frame is required for IMU state propagation. This global rotation can only be determined with known initial attitude, which is hard to obtain in many applications. IMU pre-integration, which was firstly proposed in [19], is used to handle this problem. The technique is advanced to consider on-manifold uncertainty propagation in our previous works [20, 10]. Further improvement to incorporate IMU biases and integrate with full SLAM framework is proposed in [15].

As Described in [10], if the reference frame for IMU propagation is changed to b_i, we can pre-integrate the parts in (1.33) that are related to linear acceleration $\mathbf{a}$ and angular velocity $\boldsymbol{\omega}$ as follows:

$$\begin{aligned}
\boldsymbol{\alpha}^{b_i}_{b_{i+1}} &= \iint_{t\in[i,i+1]} \boldsymbol{\gamma}^{b_i}_t \mathbf{a}_t dt^2 \\
\boldsymbol{\beta}^{b_i}_{b_{i+1}} &= \int_{t\in[i,i+1]} \boldsymbol{\gamma}^{b_i}_{b_{i+1}} \mathbf{a}_t dt \\
\boldsymbol{\gamma}^{b_i}_{b_{i+1}} &= \int_{t\in[i,i+1]} \boldsymbol{\gamma}^{b_i}_{b_{i+1}} \times \begin{bmatrix} 0 \\ \frac{1}{2}\boldsymbol{\omega}_t \end{bmatrix} dt,
\end{aligned} \tag{1.36}$$

It can be seen that the pre-integration part (1.36) can be obtained solely with IMU measurements within $[i, i+1]$ taking b_i instead of w as the base frame.

Now we are able to write down the measurement function:

$$\begin{bmatrix} \mathbf{p}^w_{b_{i+1}} \\ \mathbf{v}^w_{b_{i+1}} \\ \mathbf{q}^w_{b_{i+1}} \\ \mathbf{a}_{b_{i+1}} \\ \boldsymbol{\omega}_{b_{i+1}} \end{bmatrix} = \begin{bmatrix} \mathbf{p}^w_{b_i} + \mathbf{v}^w_{b_i}\Delta t - \frac{1}{2}\mathbf{g}\Delta t^2 + \mathbf{q}^w_{b_i}\boldsymbol{\alpha}^{b_i}_{b_{i+1}} \\ \mathbf{v}^w_{b_i} - \mathbf{g}\Delta t + \mathbf{q}^w_{b_i}\boldsymbol{\beta}^{b_i}_{b_{i+1}} \\ \mathbf{q}^w_{b_i}\boldsymbol{\gamma}^{b_i}_{b_{i+1}} \\ \mathbf{a}_{b_i} \\ \boldsymbol{\omega}_{b_i} \end{bmatrix} \tag{1.37}$$

Question 3

Although the measurement function has been written down, it is non-trivial to formulate it as a probabilistic distribution. The rotational error between $\mathbf{q}^w_{b_{i+1}}$ and $\mathbf{q}^w_{b_i}\boldsymbol{\gamma}^{b_i}_{b_{i+1}}$ lives in non-euclidean space, to be specific $SO(3)$ and is over-parameterized by quaternion. As mentioned in Sect. 1.3.3, error-state $\delta\boldsymbol{\theta}$ is employed here to model the rotational error.

With the help of logarithm operation $[\mathbf{q}]_{xyz} = [\begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix}]_{xyz} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$, the measurement error is finally defined as:

$$r_{\mathcal{B}}(\mathbf{x}_i, \mathbf{x}_{i+1}) = \begin{bmatrix} \mathbf{q}_w^{b_i}(\mathbf{p}_{b_{i+1}}^w - \mathbf{p}_{b_i}^w - \mathbf{v}_{b_i}^w \Delta t + \frac{1}{2}\mathbf{g}\Delta t^2) - \boldsymbol{\alpha}_{b_{i+1}}^{b_i} \\ \mathbf{q}_w^{b_i}(\mathbf{v}_{b_{i+1}}^w - \mathbf{v}_{b_i}^w + \mathbf{g}\Delta t) - \boldsymbol{\beta}_{b_{i+1}}^{b_i} \\ 2[\boldsymbol{\gamma}_{b_i}^{b_{i+1}} \times (\mathbf{q}_w^{b_i} \times \mathbf{q}_{b_{i+1}}^w)]_{xyz} \\ \mathbf{a}_{b_{i+1}} - \mathbf{a}_{b_i} \\ \boldsymbol{\omega}_{b_{i+1}} - \boldsymbol{\omega}_{b_i} \end{bmatrix}_{15\times 1} \tag{1.38}$$

In rest of this answer, we derive the probabilistic distribution of $r_{\mathcal{B}}(\mathbf{x}_i, \mathbf{x}_{i+1})$, which is essentially the distribution of pre-integrated part, through first-order uncertainty propagation using error-state representation

The time-variant pre-integration part $\boldsymbol{\mu}_t^{b_i} = \begin{bmatrix} \boldsymbol{\alpha}_t^{b_i} \\ \boldsymbol{\beta}_t^{b_i} \\ \boldsymbol{\gamma}_t^{b_i} \\ \mathbf{a}_t^{b_i} \\ \boldsymbol{\omega}_t^{b_i} \end{bmatrix}_{16\times 1}$ is assumed to be Gaussian in error-state representation:

$$\boldsymbol{\mu}_t^{b_i} \ominus \tilde{\boldsymbol{\mu}}_t^{b_i} = \begin{bmatrix} \boldsymbol{\alpha}_t^{b_i} - \tilde{\boldsymbol{\alpha}}_t^{b_i} \\ \boldsymbol{\beta}_t^{b_i} - \tilde{\boldsymbol{\beta}}_t^{b_i} \\ 2[\tilde{\boldsymbol{\gamma}}_{b_i}^t \times \boldsymbol{\gamma}_t^{b_i}]_{xyz} \\ \mathbf{a}_t^{b_i} - \tilde{\mathbf{a}}_t^{b_i} \\ \boldsymbol{\omega}_t^{b_i} - \tilde{\boldsymbol{\omega}}_t^{b_i} \end{bmatrix}_{15\times 1} \sim \mathcal{N}(\mathbf{0}, \Sigma_t^{b_i}). \tag{1.39}$$

Specifically, the covariance Σ_t begins with $\mathbf{0}_{15\times 15}$ at time $t = 0$ and ends with $\Sigma_t = \Sigma_{b_{i+1}}^{b_i}$ at time $t = n$, where the latter one $\Sigma_{b_{i+1}}^{b_i}$ is our final goal.

In discrete-time implementation, various numerical integration methods such as Euler, Mid-point, RK4 integration can be applied. Here the Mid-point integration is chose to demonstrate the procedure.

Suppose we have $n+1$ sampled inertial measurements, during time interval $[i, i+1]$. At k^{th} propagation, Mid-point integration updates the pre-integration part $\boldsymbol{\mu}_k^{b_i}$ using a pair of inertial measure $\begin{bmatrix} \mathbf{a}_k \\ \boldsymbol{\omega}_k \end{bmatrix}$ and $\begin{bmatrix} \mathbf{a}_{k+1} \\ \boldsymbol{\omega}_{k+1} \end{bmatrix}$:

$$\boldsymbol{\mu}_{k+1}^{b_i} = \mathrm{g}_k(\boldsymbol{\mu}_k^{b_i}, \begin{bmatrix} \mathbf{a}_k \\ \boldsymbol{\omega}_k \\ \mathbf{a}_{k+1} \\ \boldsymbol{\omega}_{k+1} \end{bmatrix}), \tag{1.40}$$

where g_k is done **in order**:

$$\boldsymbol{\omega} \leftarrow \frac{1}{2}((\boldsymbol{\omega}_k - \boldsymbol{\omega}_{b_i}) + (\boldsymbol{\omega}_{k+1} - \boldsymbol{\omega}_{b_i})) \tag{1.41}$$

$$\boldsymbol{\gamma}_{k+1}^{b_i} \leftarrow \boldsymbol{\gamma}_k^{b_i} \times \begin{bmatrix} 1 \\ \frac{1}{2}\boldsymbol{\omega} dt \end{bmatrix} \tag{1.42}$$

$$\mathbf{a} \leftarrow \frac{1}{2}(\boldsymbol{\gamma}_k^{b_i}(\mathbf{a}_k - \mathbf{a}_{b_i}) + \boldsymbol{\gamma}_{k+1}^{b_i}(\mathbf{a}_{k+1} - \mathbf{a}_{b_i})) \tag{1.43}$$

$$\boldsymbol{\alpha}_{k+1}^{b_i} \leftarrow \boldsymbol{\alpha}_k^{b_i} + \boldsymbol{\beta}_k^{b_i} dt + \frac{1}{2}\mathbf{a} dt^2 \tag{1.44}$$

$$\boldsymbol{\beta}_{k+1}^{b_i} \leftarrow \boldsymbol{\beta}_k^{b_i} + \mathbf{a} dt \tag{1.45}$$

$$\mathbf{a}_{k+1}^{b_i} \leftarrow \mathbf{a}_k^{b_i} \tag{1.46}$$

$$\boldsymbol{\omega}_{k+1}^{b_i} \leftarrow \boldsymbol{\omega}_k^{b_i} \tag{1.47}$$

Discrete-time implementation of (1.33) is to perform Mid-point integration recursively:

$$\boldsymbol{\mu}_n^{b_i} = \mathrm{g}_{n-1}(\cdots \mathrm{g}_1(\mathrm{g}_0(\boldsymbol{\mu}_0^{b_i}, \begin{bmatrix}\mathbf{a}_0\\ \boldsymbol{\omega}_0\\ \mathbf{a}_1\\ \boldsymbol{\omega}_1\end{bmatrix}), \begin{bmatrix}\mathbf{a}_1\\ \boldsymbol{\omega}_1\\ \mathbf{a}_2\\ \boldsymbol{\omega}_2\end{bmatrix}) \cdots, \begin{bmatrix}\mathbf{a}_{n-1}\\ \boldsymbol{\omega}_{n-1}\\ \mathbf{a}_n\\ \boldsymbol{\omega}_n\end{bmatrix}) \tag{1.48}$$

During the integration procedure, the estimated value of pre-integration part is propagated recursively in (1.48), so is the corresponding uncertainty. In another word, (1.48) describes how the first and second moments of the distribution (mean and covariance) evolve during integration. However, the propagation function g_i is not linear, making the first-order approximation necessary.

Taking the derivative of the nonlinear propagation function g_i gives:

$$\mathbf{A}_k = \frac{\partial \mathrm{g}_k(\boldsymbol{\mu}_k^{b_i}, \begin{bmatrix}\mathbf{a}_k\\ \boldsymbol{\omega}_k\\ \mathbf{a}_{k+1}\\ \boldsymbol{\omega}_{k+1}\end{bmatrix})}{\partial \boldsymbol{\mu}_k^{b_i}}, \mathbf{B}_k = \frac{\partial \mathrm{g}_k(\boldsymbol{\mu}_k^{b_i}, \begin{bmatrix}\mathbf{a}_k\\ \boldsymbol{\omega}_k\\ \mathbf{a}_{k+1}\\ \boldsymbol{\omega}_{k+1}\end{bmatrix})}{\partial \begin{bmatrix}\mathbf{a}_k\\ \boldsymbol{\omega}_k\\ \mathbf{a}_{k+1}\\ \boldsymbol{\omega}_{k+1}\end{bmatrix}}, \mathbf{C}_k = \begin{bmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\\ \mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}. \tag{1.49}$$

Under first-order approximation, the first and second moment are propagated as:

$$\begin{aligned}
\boldsymbol{\mu}_{k+1}^{b_i} &= \mathrm{g}_k(\boldsymbol{\mu}_k^{b_i}, \begin{bmatrix}\mathbf{a}_k\\ \boldsymbol{\omega}_k\\ \mathbf{a}_{k+1}\\ \boldsymbol{\omega}_{k+1}\end{bmatrix})\\
\Sigma_{k+1} &= \mathbf{A}_k \Sigma_k \mathbf{A}_k^\intercal\\
&\quad + \mathbf{B}_k \begin{bmatrix}\frac{\sigma_{\mathbf{a}_n}^2}{dt}\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \frac{\sigma_{\boldsymbol{\omega}_n}^2}{dt} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \frac{\sigma_{\mathbf{a}_n}^2}{dt}\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\sigma_{\boldsymbol{\omega}_n}^2}{dt}\mathbf{I}\end{bmatrix}_{12\times 12} \mathbf{B}_k^\intercal\\
&\quad + \mathbf{C}_k \begin{bmatrix}dt\sigma_{\mathbf{a}_r}^2\mathbf{I} & \mathbf{0}\\ \mathbf{0} & dt\sigma_{\boldsymbol{\omega}_r}^2\mathbf{I}\end{bmatrix}_{6\times 6} \mathbf{C}_k^\intercal
\end{aligned} \tag{1.50}$$

The sensor model defined in Sect. 1.2.2 is continuous. In our implementation we adopt discretization [21].

At time $k = n$ we need to shrink both mean and covariance to its minimal form after chaining all of the propagations between two frames:

$$\begin{aligned}
\boldsymbol{\mu}_n^{b_i} &= \mathrm{g}_{n-1}(\cdots \mathrm{g}_1(\mathrm{g}_0(\boldsymbol{\mu}_0^{b_i}, \begin{bmatrix}\mathbf{a}_0\\ \boldsymbol{\omega}_0\\ \mathbf{a}_1\\ \boldsymbol{\omega}_1\end{bmatrix}), \begin{bmatrix}\mathbf{a}_1\\ \boldsymbol{\omega}_1\\ \mathbf{a}_2\\ \boldsymbol{\omega}_2\end{bmatrix}) \cdots, \begin{bmatrix}\mathbf{a}_{n-1}\\ \boldsymbol{\omega}_{n-1}\\ \mathbf{a}_n\\ \boldsymbol{\omega}_n\end{bmatrix})\\
\mathbf{L}_{15\times 16} &= \frac{\partial \delta\boldsymbol{\mu}_n}{\partial \boldsymbol{\mu}_n} = \frac{\partial(\boldsymbol{\mu}_n \ominus \tilde{\boldsymbol{\mu}}_n)}{\partial \boldsymbol{\mu}_n}\\
\mathbf{R}_{16\times 15} &= \frac{\partial \boldsymbol{\mu}_n}{\partial \delta\boldsymbol{\mu}_0} = \frac{\partial \boldsymbol{\mu}_n}{\partial \boldsymbol{\mu}_0}\frac{\partial \boldsymbol{\mu}_0}{\partial \delta\boldsymbol{\mu}_0} = \frac{\partial(\tilde{\boldsymbol{\mu}}_0 \oplus \delta\boldsymbol{\mu}_0)}{\partial \delta\boldsymbol{\mu}_0},
\end{aligned} \tag{1.51}$$

then we have:

$$\begin{aligned} J_{b_{i+1}}^{b_i} &= \mathbf{L}\mathbf{A}_{n-1}\cdots\mathbf{A}_0\mathbf{R} \\ \mathbf{\Sigma}_{b_{i+1}}^{b_i} &= \mathbf{L}\Sigma_n\mathbf{L}^\intercal \end{aligned} \tag{1.52}$$

Intuitively, $\mathbf{L}$ describes how the small change in state vector $\boldsymbol{\mu}_n$ affects the error-state vector $\delta\boldsymbol{\mu}_n$ around the measurement $\tilde{\boldsymbol{\mu}}_n$, while $\mathbf{R}$ describes how the small change in error-state vector $\delta\boldsymbol{\mu}_0$ around the measurement $\tilde{\boldsymbol{\mu}}_0$ affects state vector $\boldsymbol{\mu}_0$.

Finally, the distribution in (1.38) is complete.

Question 4

The Jacobian:

$$J_{b_{i+1}}^{b_i} = \begin{bmatrix} \frac{\partial r_{\mathcal{B}}(\mathbf{x}_i,\mathbf{x}_{i+1})}{\partial\delta\mathbf{x}_i} & \frac{\partial r_{\mathcal{B}}(\mathbf{x}_i,\mathbf{x}_{i+1})}{\partial\delta\mathbf{x}_{i+1}} \end{bmatrix} \tag{1.53}$$

While it is trivial to evaluate the Jacobian with respect to $\begin{bmatrix}\delta\mathbf{p}\\ \delta\mathbf{v}\\ \delta\boldsymbol{\theta}\end{bmatrix}$, more considerations need to be taken for $\begin{bmatrix}\mathbf{a}_b\\ \boldsymbol{\omega}_b\end{bmatrix}$ because the pre-integration part $\boldsymbol{\mu}_{b_{i+1}}^{b_i}$ is a complex nonlinear function of $\begin{bmatrix}\mathbf{a}_{b_i}\\ \boldsymbol{\omega}_{b_i}\end{bmatrix}$. An natural way is to execute the integration procedure (1.49) in each Gauss-Newton iteration, which is obviously too computationally heavy.

Another reasonable way is to take the first-order approximation of the preintegration part $\boldsymbol{\mu}_{b_{i+1}}^{b_i}$ as:

$$\boldsymbol{\mu}_{b_{i+1}}^{b_i} = \tilde{\boldsymbol{\mu}}_{b_{i+1}}^{b_i} + \frac{\partial\boldsymbol{\mu}_{b_{i+1}}^{b_i}}{\partial\begin{bmatrix}\mathbf{a}_{b_i}\\ \boldsymbol{\omega}_{b_i}\end{bmatrix}}\begin{bmatrix}\Delta\mathbf{a}_{b_i}\\ \Delta\boldsymbol{\omega}_{b_i}\end{bmatrix} = \tilde{\boldsymbol{\mu}}_{b_{i+1}}^{b_i} + \frac{\partial\boldsymbol{\mu}_{b_{i+1}}^{b_i}}{\partial\begin{bmatrix}\mathbf{a}_{b_i}\\ \boldsymbol{\omega}_{b_i}\end{bmatrix}}\begin{bmatrix}\hat{\mathbf{a}}_{b_i}-\tilde{\mathbf{a}}_{b_i}\\ \hat{\boldsymbol{\omega}}_{b_i}-\tilde{\boldsymbol{\omega}}_{b_i}\end{bmatrix}, \tag{1.54}$$

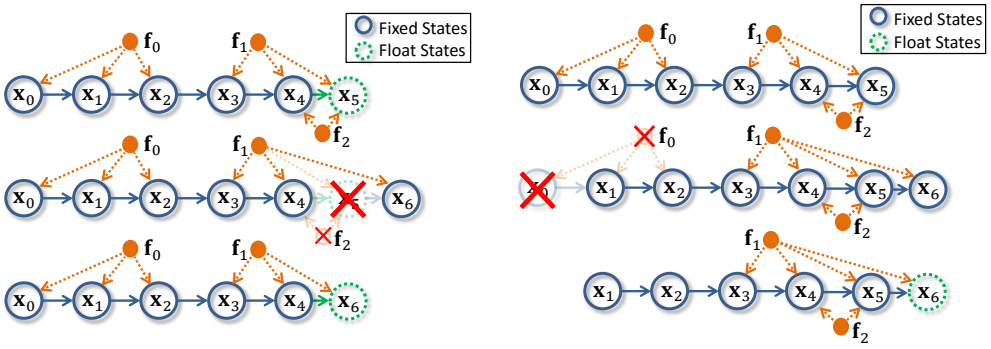
where $\begin{bmatrix}\Delta\mathbf{a}_{b_i}\\ \Delta\boldsymbol{\omega}_{b_i}\end{bmatrix}$ is the difference between latest estimation $\begin{bmatrix}\hat{\mathbf{a}}_{b_i}\\ \hat{\boldsymbol{\omega}}_{b_i}\end{bmatrix}$ and linearized point $\begin{bmatrix}\tilde{\mathbf{a}}_{b_i}\\ \tilde{\boldsymbol{\omega}}_{b_i}\end{bmatrix}$ and the Jacobian $\frac{\partial\boldsymbol{\mu}_{b_{i+1}}^{b_i}}{\partial\begin{bmatrix}\mathbf{a}_{b_i}\\ \boldsymbol{\omega}_{b_i}\end{bmatrix}}$ is the corresponding sub-matrix of the chained Jacobian $J_{b_{i+1}}^{b_i}$ obtained in (1.52). The difference $\begin{bmatrix}\Delta\mathbf{a}_{b_i}\\ \Delta\boldsymbol{\omega}_{b_i}\end{bmatrix}$ is usually small, otherwise we have to execute the integration procedure (1.49) when its norm is larger than a threshold.

In our implementation, the propagation of mean and covariance shares the same numerical integration methods. But it is not the case for other systems [4, 6], where the propagation of mean usually employs higher order integration (4^{th}-order RungeKutta methods) than the propagation of covariance (Euler methods). A possible reason is the Jacobian of the higher order integrate, which is required during covariance propagation, is to complex to write by hand. In our implementation we make use of Automatic Differentiation to evaluate (1.49). We find it affects performance little and simplified the code a lot.

1.5 Two-Way Marginalization

In order to bound the computational complexity of graph optimization, marginalization is usually used. We selectively marginalize out IMU states $\mathbf{x}_k$ and features λ_l from the factor graph (Sect. 1.3.1). Due to the well-known acceleration excitation requirement [7, 8] for scale observability for monocular VINS, a naive strategy that always marginalizes the oldest state may result in unobservable scale in degenerate motions, such as hovering or constant velocity motions.

To avoid this, we employ the two-way marginalization scheme originally proposed in our earlier works [20, 22] to selectively remove recent or old IMU states based on a scene parallax test. We add a new IMU state to the sliding window if the time between two IMU states Δt is larger than a threshold. We do not have a notion of spatial keyframes



(a) Small parallax: hovering or slow motion

(b) Large parallax: fast motion

Figure 1.2: An example with seven IMU states $\mathbf{x}_k$ and three features $\mathbf{f}_l$. Fig 1.2(a) shows the structure of the full state before, during, and after marginalizing a recent IMU state $\mathbf{x}_5$ after a newer IMU state $\mathbf{x}_6$ is added. Visual and IMU measurements related to IMU state $\mathbf{x}_5$ (denoted as transparent edges) are summarized into a new prior $\mathbf{\Lambda}_p$ in (1.55). Also, feature f_2 is removed because it has no valid observation. A similar marginalization process of the oldest IMU state is shown in Fig. 1.2(b), where IMU state $\mathbf{x}_0$ and feature f_0 are removed and all involved measurements are used to construct a new prior.

as in vision-only approaches [12], due to the requirement of bounding the uncertainty for every pre-integrated IMU measurements. We then select whether to remove the oldest or the most recent IMU states based on a parallax test. As shown in Fig. 1.2, a recent state is identified as *fixed* only if it has sufficient parallax to the previous fixed state; otherwise it will be removed in the next marginalization. We refer readers to [22] for details of this selection process. We construct a new prior based on all measurements related to the removed states:

$$\mathbf{\Lambda}_p = \mathbf{\Lambda}_p + \sum_{k \in \mathcal{B}^-} \mathbf{H}_{b_{k+1}}^{b_k \intercal} \mathbf{\Sigma}_{b_{k+1}}^{b_k \, -1} \mathbf{H}_{b_{k+1}}^{b_k} + \sum_{(l,j) \in \mathcal{C}^-} \mathbf{H}_l^{c_j \intercal} \mathbf{\Sigma}_l^{c_j \, -1} \mathbf{H}_l^{c_j}, \tag{1.55}$$

where $\mathcal{B}^-$ and $\mathcal{C}^-$ are sets of removed IMU and camera measurements respectively. The marginalization is carried out using the Schur complement [11].

Intuitively, the two-way marginalization keeps removing recent IMU states if the platform has small or no motion. Keeping older IMU states in this case will preserve acceleration information that is necessary to recover the visual scale. In the next chapter we will see the camera-IMU calibration also benefits from this marginalization scheme because it naturally accumulates all measurements to refine the calibration parameters.

1.6 Experimental Results

1.6.1 Implementation Details

As shown in Fig. 1.3, our monocular VINS sensor suite consists of an mvBlueFOX-MLC200w grayscale HDR camera with a wide-angle lens that captures 752×480 images at 30 Hz and a Microstrain 3DM-GX4 IMU that runs at 500 Hz. The mount for the sensor suite has significant translation between sensors. The sensors are also purposely mounted in different frames with approximately 90 degree rotation offsets in roll and yaw to test the performance of camera-IMU calibration.

Our algorithm runs real-time on an Intel NUC mini PC with i5-4250U processor. Three threads run in parallel in our implementation. The first thread performs detection of corner features [23] at 10 Hz and KLT tracking [24] at 30 Hz. The second thread performs initialization and calibration, as well as nonlinear optimization at 10 Hz. We maintain $N = 30$ IMU states (30 images) and $M = 200$ features in the sliding window. For each image, we detect a maximum

Figure 1.3: Our monocular VINS setup. The camera-IMU extrinsic transformation is calibrated.

of 100 new features with a minimum separation of 30 pixels. A tracked feature has to pass a rotation-compensated parallax threshold of 30 pixels before it can be triangulated and added into the optimization.

1.6.2 Performance in Large Scale Environments

In this experiment, we evaluate the performance of the overall system with challenging datasets in complex indoor and outdoor environments.

In the first scenario, we hand-hold the sensor suite and move it in a complex library environment. We encounter large rotation (Fig. 1.4(a)), motion blur (Fig. 1.4(b)), people walking and view obstruction (Fig. 1.4(c)), as well as mirrors (Fig. 1.4(d)). Fig. 1.5 shows the position estimation from the overall monocular VINS estimator with challenging cases shown in Fig. 1.4. The total trajectory length is 247.36 meters and the final position drift is 3.28 meters. The error is $1.3\,\%$ of the total trajectory length. However, considering that during the experiment the sensor suite reaches an angular velocity up to 120 degree/s, which causes significant motion blur, we can still claim that overall estimation accuracy is high.

In the second scenario, we consider aerial navigation tasks in complex outdoor environments. We mount our sensor suite on a quadrotor to show its capability to assist autonomous navigation. Our system setup consists of a forward-facing camera, a downward-facing camera, and an IMU. All cameras capture data at the same time, which allows us to evaluate the performance of the proposed system with two configurations: forward-facing camera+IMU and downward-facing camera+IMU. The testing site spans a variety of cases, including narrow sidewalk (Fig. 1.6(a)), open space (Fig. 1.6(b)), high-speed flight (Fig. 1.6(c)), and large rotation (Fig. 1.6(d)). The total flight time is approximately 8 minutes, and the vehicle travels 642 meters with a highest speed of 8.9 m/s and a height range of $[-6.6, 4.5]$ (units in meters). The trajectory is aligned with an aerial map using GPS measurements as position reference (Fig. 1.7). Note that GPS reference is only available when the quadrotor is far away from buildings.

We run the system pipeline separately with the two configurations. As shown in Fig. 1.7, the final position drifts obtained with the forward- and downward-facing configurations are 5.86 m and 2.80 m respectively, which correspond to $0.91\,\%$ and $0.44\,\%$ of the total trajectory length. While Fig. 1.7 reports that the downward-facing version aligns better with GPS in the $x - y$ direction, we observe that the downward-facing camera accumulates more drift in the z direction (1.75 meters). This result makes sense since it is well known that the estimation performance is worse when the direction of movement is parallel to the camera's optical axis. Another reason that the forward-facing configuration

(a) Large rotation (b) Motion blur

(c) People walking (d) Mirror

Figure 1.4: Onboard images during experiment in indoor environments.

performs worse is that the camera often observes only distant features (Fig. 1.6(b)), resulting in no features being triangulated for an extended period of time (Sect. 1.6.1). The performance difference indicates the necessity of choosing different sensor configurations to adapt to different applications.

1.7 Conclusion

In this chapter, we give a tutorial about our implementation of monocular VINS. The information from camera and IMU is fused together based on probabilistic graphical model. Its efficiency, accuracy and robustness is verified through comprehensive experiments in complex indoor and outdoor.

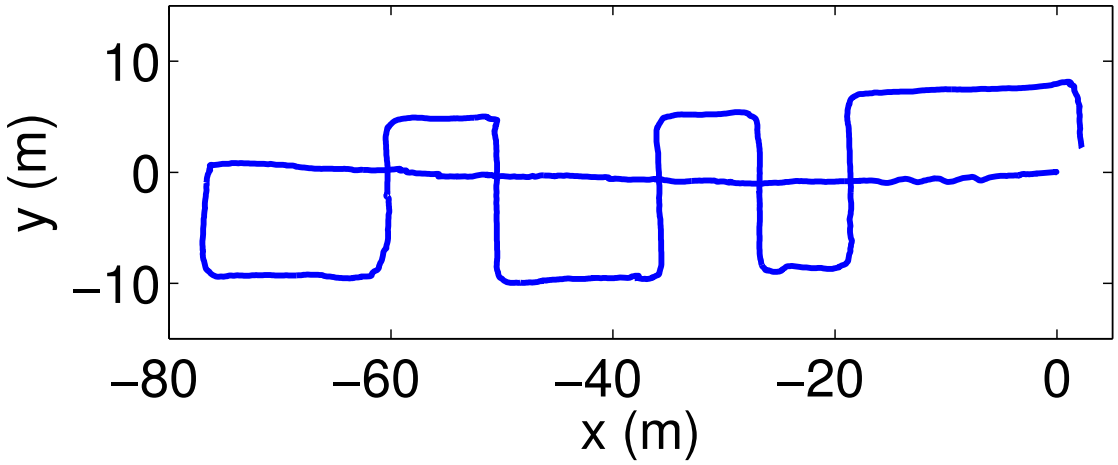


Figure 1.5: Position estimation with our monocular VINS estimator in a complex indoor environment. The total trajectory length is 247.36 m and the final position drift is 3.28 m.

(a) Narrow sidewalk

(b) Hovering in open space (4.5 m above the ground)

(c) Low altitude high speed flight (highest velocity 8.9 m/s)

(d) Aggressive braking (largest pitch 35 degrees)

Figure 1.6: Onboard images during experiment in outdoor environment. For each case, two images, one captured by the forward-facing (left) and the other one by the downward-facing camera (right), are shown.

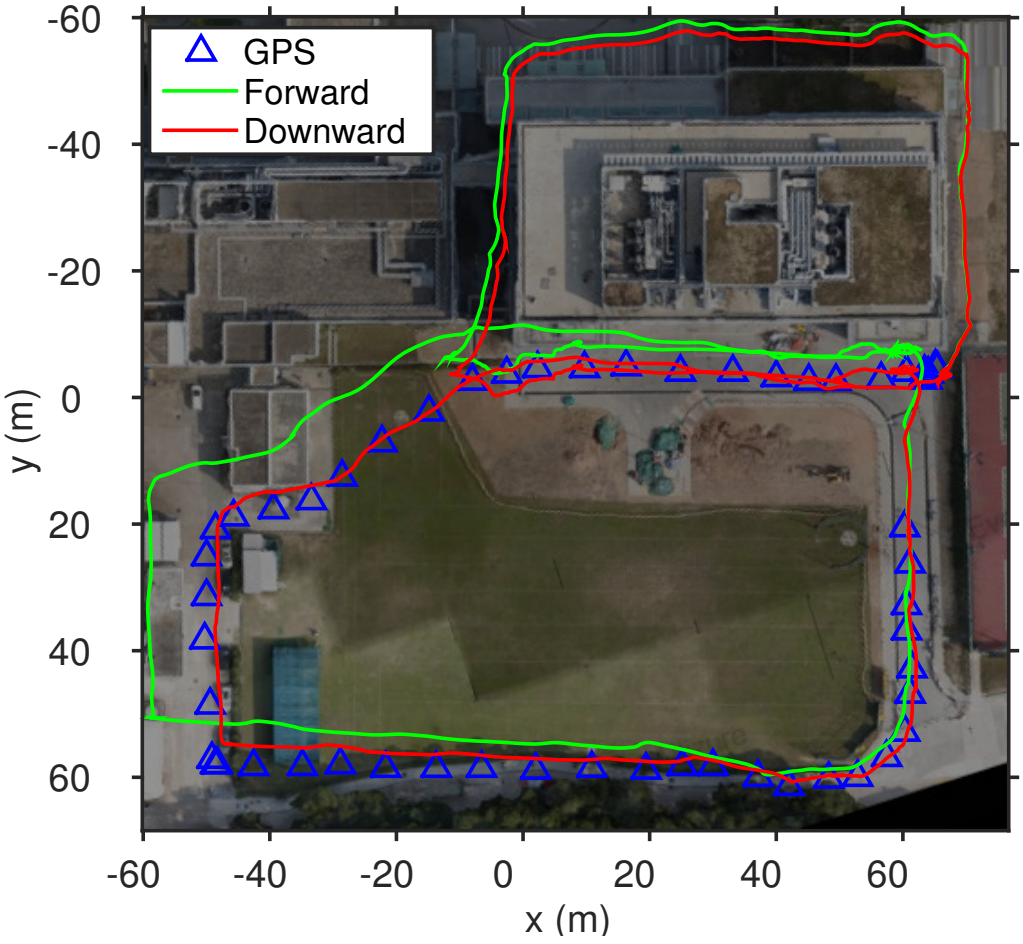


Figure 1.7: Comparison between different sensor configurations (forward- or downward-facing monocular camera) in the same flight. The total travel distance is $642\,\text{m}$. Final position drifts are $0.91\,\%$ and $0.44\,\%$ of the traveled distance for the two configurations respectively. We observe that the forward-facing camera accumulates more drift in the $x-y$ direction, while the downward-facing camera accumulates more drift in the z direction (1.75 meters). This shows the necessity of rapid mission-dependent sensor reconfiguration.

Appendix A

Some Appendix

A.1 Quaternion

A.1.1 Definition

$$\mathbf{q} \triangleq \begin{bmatrix} q_w \\ \overrightarrow{\mathbf{q}} \end{bmatrix}$$

$$\bar{\mathbf{x}} \triangleq \begin{bmatrix} \mathbf{0} \\ \overrightarrow{\mathbf{x}} \end{bmatrix}$$

$$\mathbf{q}^* \triangleq \begin{bmatrix} q_w \\ -\overrightarrow{\mathbf{q}} \end{bmatrix}$$

$$[\mathbf{q}]_{xyz} \triangleq \overrightarrow{\mathbf{q}}$$

A.1.2 Composition through *linear* Matrix Multiplication

$$\mathbf{q} \otimes \mathbf{p} = \mathbf{L}(\mathbf{q})\mathbf{p} = \begin{bmatrix} q_w & -\overrightarrow{\mathbf{q}}^\top \\ \overrightarrow{\mathbf{q}} & q_w\mathbf{I} + \hat{\overrightarrow{\mathbf{q}}} \end{bmatrix} \mathbf{p}$$

$$\mathbf{L}(\mathbf{q}) = \mathbf{L}^\intercal(\mathbf{q}^*)$$

$$\mathbf{q} \otimes \mathbf{p} = \mathbf{R}(\mathbf{p})\mathbf{q} = \begin{bmatrix} p_w & -\overrightarrow{\mathbf{p}}^\intercal \\ \overrightarrow{\mathbf{p}} & p_w\mathbf{I} + \hat{\overrightarrow{\mathbf{p}}} \end{bmatrix} \mathbf{q}$$

$$\mathbf{R}(\mathbf{p}) = \mathbf{R}^\intercal(\mathbf{p}^*)$$

A.1.3 Commutative Law

Because

$$\mathbf{L}(\mathbf{q})\mathbf{R}(\mathbf{p})\mathbf{r} = \mathbf{q} \otimes \mathbf{r} \otimes \mathbf{p} = \mathbf{R}(\mathbf{p})\mathbf{L}(\mathbf{q})\mathbf{r},$$

we have

$$\mathbf{L}(\mathbf{q})\mathbf{R}(\mathbf{p}) = \mathbf{R}(\mathbf{p})\mathbf{L}(\mathbf{q}).$$

A.1.4 Rotate a Vector

$$\begin{aligned}\mathbf{q} \times \mathbf{x} &\triangleq \mathcal{R}(\mathbf{q})\mathbf{x} \\ &= \mathbf{q} \otimes \bar{\mathbf{x}} \otimes \mathbf{q}^* \\ &= \mathbf{x} + 2\overrightarrow{\mathbf{q}} \times (\overrightarrow{\mathbf{q}} \times \mathbf{x} + q_w \mathbf{x}) \\ &= \mathbf{x} + 2\left[\overrightarrow{\mathbf{q}}\right]_\times^2 \mathbf{x} + 2q_w \left[\overrightarrow{\mathbf{q}}\right]_\times \mathbf{x} \\ &= \mathbf{x} + 2(\overrightarrow{\mathbf{q}}\,\overrightarrow{\mathbf{q}}^\intercal - \mathbf{I})\mathbf{x} + 2q_w \left[\overrightarrow{\mathbf{q}}\right]_\times \mathbf{x}\end{aligned}$$

A.1.5 Conversion to Rotation Matrix

$$\begin{aligned}\mathbf{q} \otimes \bar{\mathbf{x}} \otimes \mathbf{q}^* &= \mathbf{L}(\mathbf{q})\mathbf{R}(\mathbf{q}^*)\bar{\mathbf{x}} \\ \begin{bmatrix}\mathbf{0} \\ \mathcal{R}\overrightarrow{\mathbf{x}}\end{bmatrix} &= \mathbf{L}(\mathbf{q})\mathbf{R}(\mathbf{q}^*)\begin{bmatrix}\mathbf{0} \\ \overrightarrow{\mathbf{x}}\end{bmatrix} \\ \mathcal{R}(\mathbf{q}) &= \begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix}\mathbf{L}(\mathbf{q})\mathbf{R}(\mathbf{q}^*)\begin{bmatrix}\mathbf{0} \\ \mathbf{I}\end{bmatrix}\end{aligned}$$

A.1.6 Covariance Propagation of Process Noise

Discrete-time System

$$\begin{aligned}x_k &= x_{k-1} + w_{k-1} \\ w_k &\simeq (0, Q) \\ x_0 &= 0\end{aligned}$$

The discrete-time system can be solved as follows:

$$x_k = w_0 + w_1 + \cdots + w_{k-1}$$

, and the covariance of the state is therefore given as:

$$\begin{aligned}E[x_k x_k^\intercal] &= E[(w_0 + w_1 + \cdots + w_{k-1})(w_0 + w_1 + \cdots + w_{k-1})^\intercal] \\ &= E[x_0 x_0^\intercal] + E[x_1 x_1^\intercal] + \cdots + E[x_{k-1} x_{k-1}^\intercal] \\ &= kQ\end{aligned}$$

Continuous-time System

$$\begin{aligned}\dot{x}(t) &= w(t) \\ w(t) &\simeq (0, Q) \\ x(0) &= 0\end{aligned}$$

We **propose** the following definition for continuous-time white noise:

$$E[w(t)w(\tau)^\intercal] = \frac{Q}{T}\delta(t - \tau)$$

. Then we are able to compute the covariance of $x(t)$:

$$\begin{aligned} E[x(t)x^\intercal(t)] &= E[\int_0^t w(\alpha)d\alpha \int_0^t w^\intercal(\beta)d\beta] \\ &= \int_0^t \int_0^t E[w(\alpha)w^\intercal(\beta)]d\alpha d\beta \\ &= \int_0^t \int_0^t \frac{Q}{T}\delta(t-\tau)d\alpha d\beta \\ &= \int_0^t \frac{Q}{T}d\beta \\ &= \frac{Qt}{T} \end{aligned}$$

Recalling that $t = kT$, we can write the above equation as:

$$E[x(t)x^\intercal(t)] = kQ$$

.

The above equation implies continuous-time system with white noise

$$w(t) \simeq (0, Q_c), Q_c = \frac{Q}{T}$$

is **equivalent** to discrete-time system with noise

$$w_k \simeq (0, Q)$$

.

Finally, we are able to predict the covariance of $x(t)$ in the same way to that we used in discrete-time system:

$$E[x(t)x^\intercal(t)] = kQ = kQ_cT = tQ_c$$

, which is regardless of sample time T.

A.2 Formulation

A.2.1 IMU Model

$$\begin{aligned} \mathbf{acc}_t &= \mathbf{acc} + \mathbf{na} + \mathbf{ba} \\ \mathbf{gyr}_t &= \mathbf{gyr} + \mathbf{ng} + \mathbf{bg} \end{aligned}$$

A Gaussian white noise $\mathbf{n}$ with standard derivation σ is defined as:

$$\begin{aligned} \mathbf{E}[\mathbf{n}] &= \mathbf{0}_{3\times 1} \\ \mathbf{E}[\mathbf{n}(t+\tau)\mathbf{n}^\intercal(t)] &= \delta(\tau)\sigma^2\mathbf{I}_{3\times 3} \end{aligned}$$

We assume $\mathbf{na}, \mathbf{ng}, \dot{\mathbf{ba}}, \dot{\mathbf{bg}}$ are Gaussian white noise with standard derivation $\sigma_{na}, \sigma_{ng}, \sigma_{wa}, \sigma_{wg}$.
In discrete implementation, we get:
At time t, sampled $\mathbf{na}_t, \mathbf{ng}_t$ follow Gaussian distribution: $\mathbf{na}_t \sim \mathcal{N}(\mathbf{0}, \sigma_{na}^2\mathbf{I}), \mathbf{ng}_t \sim \mathcal{N}(\mathbf{0}, \sigma_{ng}^2\mathbf{I})$.
During time interval $[t, t+dt]$, Integrated $\Delta\mathbf{ba}_t = \int_t^{t+dt} \dot{\mathbf{ba}}\mathrm{d}t$, $\Delta\mathbf{bg}_t = \int_t^{t+dt} \dot{\mathbf{bg}}\mathrm{d}t$ follow Gaussian distribution: $\Delta\mathbf{ba}_t \sim \mathcal{N}(\mathbf{0}, (\sigma_{na}\sqrt{dt})^2\mathbf{I}), \Delta\mathbf{bg}_t \sim \mathcal{N}(\mathbf{0}, (\sigma_{ng}\sqrt{dt})^2\mathbf{I})$.
The latter one follows theory in stochastic integration.

A.2.2 PreIntegration

In this part, the goal is:

1. propagate the *mean*.
2. propagate the *covariance*.
3. propagate the *Jacobian*.

First, we define:

$$\mathbf{x}_n = \begin{bmatrix} \alpha_n \\ \beta_n \\ \mathbf{q}_n \\ \mathbf{ba}_n \\ \mathbf{bg}_n \end{bmatrix}_{16\times 1}, \mathbf{x}_n \sim \mathcal{N}(\mu_n, \Sigma_n)$$

$$\mathbf{x}_{n+1} = f(\mathbf{x}_n, dt, \mathbf{acc}_n, \mathbf{gyr}_n, \mathbf{acc}_{n+1}, \mathbf{gyr}_{n+1})$$

$$\mathbf{J}_n = \frac{\partial \mathbf{x}_n}{\partial \mathbf{x}_0}$$

where $f()$ could be Euler, Midpoint or RK4 integration.

Begin with

$$\mu_0 = \begin{bmatrix} \mathbf{0}_{3\times 1} \\ \mathbf{0}_{3\times 1} \\ \begin{bmatrix} 1 \\ \mathbf{0}_{3\times 1} \end{bmatrix} \\ \mathbf{0}_{3\times 1} \\ \mathbf{0}_{3\times 1} \end{bmatrix}_{16\times 1}, \Sigma_0 = \mathbf{0}_{16\times 16}, \mathbf{J}_0 = \mathbf{I}_{16\times 16}$$

the distribution is propagated recursively:

$$\mu_{n+1} = f(\mu_n, dt, \mathbf{acc}_n, \mathbf{gyr}_n, \mathbf{acc}_{n+1}, \mathbf{gyr}_{n+1})$$

$$\mathbf{A}_{16\times 16} = \frac{\partial f}{\partial \mathbf{x}}, \mathbf{B}_{16\times 12} = \begin{bmatrix} \frac{\partial f}{\partial \mathbf{acc}_n} & \frac{\partial f}{\partial \mathbf{gyr}_n} & \frac{\partial f}{\partial \mathbf{acc}_{n+1}} & \frac{\partial f}{\partial \mathbf{gyr}_{n+1}} \end{bmatrix}, \mathbf{C}_{16\times 6} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

$$\Sigma_{n+1} = \mathbf{A}\Sigma_n\mathbf{A}^\intercal + \mathbf{B} \begin{bmatrix} \sigma_{na}^2\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \sigma_{nw}^2\mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \sigma_{na}^2\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \sigma_{ng}^2\mathbf{I} \end{bmatrix}_{12\times 12} \mathbf{B}^\intercal + \mathbf{C} \begin{bmatrix} dt\sigma_{wa}^2\mathbf{0} & \mathbf{0} \\ \mathbf{0} & dt\sigma_{wg}^2\mathbf{0} \end{bmatrix}_{6\times 6} \mathbf{C}^\intercal$$

$$\mathbf{J}_{n+1} = \mathbf{A}\mathbf{J}_n$$

Finally, we need to shrink both mean and covariance to its minimal form. After chaining all of the propagations between two keyframes, define

$$\begin{aligned} \mathbf{x}_{N-1} &= g(\mathbf{x}_0, dt, \mathbf{acc}_{0\ldots N-1}, \mathbf{gyr}_{0\ldots N-1}) \\ \mathbf{L}_{15\times 16} &= \frac{\partial \delta\mathbf{x}_{N-1}}{\partial \mathbf{x}_{N-1}} = \frac{\partial (\mathbf{x}_{N-1} \ominus \mu_{N-1})}{\partial \mathbf{x}_{N-1}} \\ \mathbf{R}_{16\times 15} &= \frac{\partial \mathbf{x}_{N-1}}{\partial \delta\mathbf{x}_0} = \frac{\partial \mathbf{x}_{N-1}}{\partial \mathbf{x}_0}\frac{\partial \mathbf{x}_0}{\partial \delta\mathbf{x}_0} = \mathbf{J}_{N-1}\frac{\partial (\mu_0 \oplus \delta\mathbf{x}_0)}{\partial \delta\mathbf{x}_0}, \end{aligned}$$

then we have:

$$\mathbf{J}_{minimal} = \mathbf{L}\mathbf{J}_{N-1}\mathbf{R}$$
$$\Sigma_{minimal} = \mathbf{L}\Sigma_{N-1}\mathbf{L}^{\intercal}$$

A.2.3 MLE Formulation

$$\mathbf{residual}_{ij} = \begin{bmatrix} \mathbf{Ep}_{ij} \\ \mathbf{Ev}_{ij} \\ \mathbf{Eq}_{ij} \\ \mathbf{Eba}_{ij} \\ \mathbf{Ebg}_{ij} \end{bmatrix} = \begin{bmatrix} \mathbf{q}_i^* \times (\frac{1}{2}\mathbf{g}dt^2 + \mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i dt) - \alpha_{ij} \\ \mathbf{q}_i^* \times (\mathbf{g}dt + \mathbf{v}_j - \mathbf{v}_i) - \beta_{ij} \\ 2\left[\Delta\mathbf{q}_{ij}^* \otimes \mathbf{q}_i^* \otimes \mathbf{q}_j\right]_{xyz} \\ \mathbf{ba}_j - \mathbf{ba}_i \\ \mathbf{bg}_j - \mathbf{bg}_i \end{bmatrix}$$

The propagated mean and covariance can be used to define the cost function based on MLE.
The propagated Jacobian is required by Gauss-Newton optimization.
So-called PreIntegration method is doable after observing the block elements of the Jacobian.

A.3 Analytic Jacobian

A.3.1 With respect to Bias

A.3.2 With respect to Rotation

$\frac{\partial \mathbf{r}_{ij}}{\partial \theta_j}$

$$\begin{aligned}
\frac{\partial \mathbf{r}_{ij}}{\partial \theta_j} &= \frac{\partial 2\left[\Delta\mathbf{q}_{ij}^* \otimes \mathbf{q}_i^* \otimes \mathbf{q}_j \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\theta_j \end{bmatrix}\right]_{xyz}}{\partial \theta_j} \\
&= 2\begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix} \frac{\partial(\Delta\mathbf{q}_{ij}^* \otimes \mathbf{q}_i^* \otimes \mathbf{q}_j \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\theta_j \end{bmatrix})}{\partial \theta_j} \\
&= 2\begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix} \frac{\partial(\mathbf{L}(\Delta\mathbf{q}_{ij}^* \otimes \mathbf{q}_i^* \otimes \mathbf{q}_j) \begin{bmatrix} 1 \\ \frac{1}{2}\theta_j \end{bmatrix})}{\partial \theta_j} \\
&= 2\begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix} \frac{\partial(\mathbf{L}(\Delta\mathbf{q}_{ij}^* \otimes \mathbf{q}_i^* \otimes \mathbf{q}_j) \begin{bmatrix} 1 \\ \frac{1}{2}\theta_j \end{bmatrix})}{\partial \begin{bmatrix} 1 \\ \frac{1}{2}\theta_j \end{bmatrix}} \frac{\partial \begin{bmatrix} 1 \\ \frac{1}{2}\theta_j \end{bmatrix}}{\partial \theta_j} \\
&= 2\begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix} (\mathbf{L}(\Delta\mathbf{q}_{ij}^* \otimes \mathbf{q}_i^* \otimes \mathbf{q}_j)) \begin{bmatrix} 1 \\ \frac{1}{2}\mathbf{I} \end{bmatrix} \\
&\approx \mathbf{I}
\end{aligned}$$

Given

$$\Delta\mathbf{q}_{ij} \approx \mathbf{q}_i^* \otimes \mathbf{q}^j,$$

we have

$$\frac{\partial \mathbf{r}_{ij}}{\partial \theta_j} \approx \mathbf{I}$$

A.3.3 $\frac{\partial \mathbf{r}_{ij}}{\partial \theta_i}$

$$
\begin{aligned}
\frac{\partial \mathbf{r}_{ij}}{\partial \theta_i} &= \frac{\partial 2 \left[\Delta \mathbf{q}_{ij}^* \otimes [\mathbf{q}_i \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\theta_i \end{bmatrix}]^* \otimes \mathbf{q}_j \right]_{xyz}}{\partial \theta_i} \\
&= \frac{\partial -2 \left[[\Delta \mathbf{q}_{ij}^* \otimes [\mathbf{q}_i \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\theta_i \end{bmatrix}]^* \otimes \mathbf{q}_j]^* \right]_{xyz}}{\partial \theta_i} \\
&= \frac{\partial -2 \left[[\mathbf{q}_j^* \otimes [\mathbf{q}_i \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\theta_i \end{bmatrix}] \otimes \Delta \mathbf{q}_{ij} \right]_{xyz}}{\partial \theta_i} \\
&= -2 \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \frac{\partial [\mathbf{q}_j^* \otimes [\mathbf{q}_i \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\theta_i \end{bmatrix}] \otimes \Delta \mathbf{q}_{ij}]}{\partial \theta_i} \\
&= -2 \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \frac{\partial [\mathbf{q}_j^* \otimes \mathbf{q}_i \otimes [\begin{bmatrix} 1 \\ \frac{1}{2}\theta_i \end{bmatrix} \otimes \Delta \mathbf{q}_{ij}]]}{\partial \theta_i} \\
&= -2 \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \frac{\partial [\mathbf{q}_j^* \otimes \mathbf{q}_i \otimes [\mathbf{R}(\Delta \mathbf{q}_{ij}) \begin{bmatrix} 1 \\ \frac{1}{2}\theta_i \end{bmatrix}]]}{\partial \theta_i} \\
&= -2 \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \frac{\partial [\mathbf{L}(\mathbf{q}_j^* \otimes \mathbf{q}_i) \mathbf{R}(\Delta \mathbf{q}_{ij}) \begin{bmatrix} 1 \\ \frac{1}{2}\theta_i \end{bmatrix}]}{\partial \theta_i} \\
&= -2 \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \frac{\partial [\mathbf{L}(\mathbf{q}_j^* \otimes \mathbf{q}_i) \mathbf{R}(\Delta \mathbf{q}_{ij}) \begin{bmatrix} 1 \\ \frac{1}{2}\theta_i \end{bmatrix}]}{\partial \begin{bmatrix} 1 \\ \frac{1}{2}\theta_i \end{bmatrix}} \frac{\partial \begin{bmatrix} 1 \\ \frac{1}{2}\theta_i \end{bmatrix}}{\partial \theta_i} \\
&= -2 \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} [\mathbf{L}(\mathbf{q}_j^* \otimes \mathbf{q}_i) \mathbf{R}(\Delta \mathbf{q}_{ij})] \begin{bmatrix} 1 \\ \frac{1}{2}\mathbf{I} \end{bmatrix}
\end{aligned}
$$

Given

$$\Delta \mathbf{q}_{ij} \approx \mathbf{q}_i^* \otimes \mathbf{q}^j,$$

we have

$$\frac{\partial \mathbf{r}_{ij}}{\partial \theta_i} \approx -\mathcal{R}(\Delta \mathbf{q}_{ij}^*)$$

A.3.4 $\frac{\partial \mathbf{r}_{ij}}{\partial \theta_{ij}}$

$$\begin{aligned}
\frac{\partial \mathbf{r}_{ij}}{\partial \theta_i} &= \frac{\partial 2 \left[[\Delta \mathbf{q}_{ij} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}\mathbf{J}\Delta\theta_{ij} \end{bmatrix}]^* \otimes \mathbf{q}_i^* \otimes \mathbf{q}_j \right]_{xyz}}{\partial \Delta\theta_{ij}} \\
&= \frac{\partial -2 \left[[[\Delta \mathbf{q}_{ij} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}\mathbf{J}\Delta\theta_{ij} \end{bmatrix}]^* \otimes \mathbf{q}_i^* \otimes \mathbf{q}_j]^* \right]_{xyz}}{\partial \Delta\theta_{ij}} \\
&= \frac{\partial -2 \left[\mathbf{q}_j^* \otimes \mathbf{q}_i \otimes [\Delta \mathbf{q}_{ij} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}\mathbf{J}\Delta\theta_{ij} \end{bmatrix}] \right]_{xyz}}{\partial \Delta\theta_{ij}} \\
&= -2 \begin{bmatrix} 0 & \mathbf{I} \end{bmatrix} \frac{\partial [\mathbf{q}_j^* \otimes \mathbf{q}_i \otimes [\Delta \mathbf{q}_{ij} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}\mathbf{J}\Delta\theta_{ij} \end{bmatrix}]]}{\partial \Delta\theta_{ij}} \\
&= -2 \begin{bmatrix} 0 & \mathbf{I} \end{bmatrix} \mathbf{L}(\mathbf{q}_j^* \otimes \mathbf{q}_i \otimes \Delta \mathbf{q}_{ij}) \begin{bmatrix} 0 \\ \frac{1}{2}\mathbf{J} \end{bmatrix}
\end{aligned}$$

Given

$$\Delta \mathbf{q}_{ij} \approx \mathbf{q}_i^* \otimes \mathbf{q}^j,$$

we have

$$\frac{\partial \mathbf{r}_{ij}}{\partial \theta_{ij}} \approx \mathbf{J}$$

A.3.5 With respect to Velocity

A.3.6 With respect to Position

A.4 Remaining Problems

Need we positify quaternion?

Bibliography

[1] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Consistency analysis and improvement of vision-aided inertial navigation," *IEEE Trans. Robot.*, vol. 30, no. 1, pp. 158–176, Feb. 2014.

[2] M. Li and A. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *Int. J. Robot. Research*, vol. 32, no. 6, pp. 690–711, May 2013.

[3] D. Scaramuzza, M. Achtelik, L. Doitsidis, F. Fraundorfer, E. Kosmatopoulos, A. Martinelli, M. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip, D. Gurdan, L. Heng, G. Lee, S. Lynen, L. Meier, M. Pollefeys, A. Renzaglia, R. Siegwart, J. Stumpf, P. Tanskanen, C. Troiani, and S. Weiss, "Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in GPS-denied environments," *IEEE Robot. Autom. Mag.*, vol. 21, no. 3, 2014.

[4] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, "Keyframe-based visual-inertial SLAM using nonlinear optimization," in *Proc. of Robot.: Sci. and Syst.*, Berlin, Germany, Jun. 2013.

[5] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Proc. of the Int. Sym. of Robot. Research*, Flagstaff, AZ, Aug. 2011.

[6] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proc. of the IEEE Int. Conf. on Robot. and Autom.*, Roma, Italy, Apr. 2007, pp. 3565–3572.

[7] J. Kelly and G. S. Sukhatme, "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration," *Int. J. Robot. Research*, vol. 30, no. 1, pp. 56–79, Jan. 2011.

[8] E. S. Jones and S. Soatto, "Visual-inertial navigation, mapping and localization: A scalable real-time causal approach," *Int. J. Robot. Research*, vol. 30, no. 4, pp. 407–430, Apr. 2011.

[9] V. Indelman, S. Williams, M. Kaess, and F. Dellaert, "Information fusion in navigation systems via factor graph based incremental smoothing," *Robot. and Auton. Syst.*, vol. 61, no. 8, pp. 721–738, 2013.

[10] S. Shen, N. Michael, and V. Kumar, "Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft MAVs," in *Proc. of the IEEE Int. Conf. on Robot. and Autom.*, Seattle, WA, May 2015.

[11] G. Sibley, L. Matthies, and G. Sukhatme, "Sliding window filter with application to planetary landing," *J. Field Robot.*, vol. 27, no. 5, pp. 587–608, Sep. 2010.

[12] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. of the IEEE and ACM Int. Sym. on Mixed and Augmented Reality*, Nara, Japan, Nov. 2007.

[13] S. Lange, N. Sunderhauf, and P. Protzel, "Incremental smoothing vs. filtering for sensor fusion on an indoor UAV," in *Proc. of the IEEE Int. Conf. on Robot. and Autom.*, Karlsruhe, Germany, May 2013, pp. 1773–1778.

[14] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments," in *Proc. of the IEEE Int. Conf. on Robot. and Autom.*, Saint Paul, MN, May 2012, pp. 957–964.

[15] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," in *Proc. of Robot.: Sci. and Syst.*, Rome, Italy, Jul. 2015.

[16] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proc. of the IEEE Int. Conf. on Robot. and Autom.*, Hong Kong, China, May 2014.

[17] P. Huber, "Robust estimation of a location parameter," *Annals of Mathematical Statistics*, vol. 35, no. 2, pp. 73–101, 1964.

[18] M. I. Lourakis and A. A. Argyros, "Is levenberg-marquardt the most efficient optimization algorithm for implementing bundle adjustment?" in *Proc. of the IEEE Int. Conf. Comput. Vis.*, vol. II, 2005, pp. 1526–1531.

[19] T. Lupton and S. Sukkarieh, "Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions," *IEEE Trans. Robot.*, vol. 28, no. 1, pp. 61–76, Feb. 2012.

[20] S. Shen, "Autonomous navigation in complex indoor and outdoor environments with micro aerial vehicles," Ph.D. dissertation, University of Pennsylvania, Philadelphia, PA, Aug. 2014.

[21] J. L. Crassidis, "Sigma-point kalman filtering for integrated gps and inertial navigation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, no. 2, pp. 750–756, Apr. 2006.

[22] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Initialization-free monocular visual-inertial estimation with application to autonomous MAVs," in *Proc. of the Int. Sym. on Exp. Robot.*, Marrakech, Morocco, Jun. 2014.

[23] J. Shi and C. Tomasi, "Good features to track," in *Proc. of the IEEE Int. Conf. on Pattern Recognition*, Seattle, WA, Jun. 1994, pp. 593–600.

[24] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. of the Int. Joint Conf. on Artificial Intelligence*, Vancouver, Canada, Aug. 1981, pp. 24–28.

[25] L. Kneip, S. Weiss, and R. Siegwart, "Deterministic initialization of metric state estimation filters for loosely-coupled monocular vision-inertial systems," in *Proc. of the IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, San Francisco, CA, Sep. 2011, pp. 2235–2241.

[26] T. Dong-Si and A. I. Mourikis, "Estimator initialization in vision-aided inertial navigation with unknown camera-IMU calibration," in *Proc. of the IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, Vilamoura, Algarve, Portugal, Oct. 2012, pp. 1064–1071.

[27] A. Martinelli, "Vision and IMU data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination," *IEEE Trans. Robot.*, vol. 28, no. 1, pp. 44–60, Feb. 2012.

[28] ——, "Closed-form solution of visual-inertial structure from motion," *Int. J. Comput. Vis.*, vol. 106, no. 2, pp. 138–152, 2014.

[29] V. Lippiello and R. Mebarki, "Closed-form solution for absolute scale velocity estimation using visual and inertial data with a sliding least-squares estimation," in *Proc. of Mediterranean Conf. on Control and Automation*, Platanias-Chania, Crete, Greece, Jun. 2013, pp. 1261–1266.

[30] L. Heng, G. H. Lee, and M. Pollefeys, "Self-calibration and visual SLAM with a multi-camera system on a micro aerial vehicle," in *Proc. of Robot.: Sci. and Syst.*, Berkeley, CA, Jul. 2014.

[31] D. Nister, "An efficient solution to the five-point relative pose problem," in *Proc. of the IEEE Int. Conf. on Pattern Recognition*, vol. 2, Madison, WI, Jun. 2003, pp. 195–202.

[32] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. 32, no. 1, pp. 105–119, Jan. 2010.

[33] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *Proc. of the IEEE and ACM Int. Sym. on Mixed and Augmented Reality*, Oct. 2011, pp. 127–136.

[34] V. Pradeep, C. Rhemann, S. Izadi, C. Zach, M. Bleyer, and S. Bathiche, "MonoFusion: Real-time 3D reconstruction of small scenes with a single web camera," in *Proc. of the IEEE and ACM Int. Sym. on Mixed and Augmented Reality*, Oct. 2013, pp. 83–88.

[35] P. Ondruska, P. Kohli, and S. Izadi, "MobileFusion: Real-time volumetric surface reconstruction and dense tracking on mobile phones," *IEEE Trans. Vis. Comput. Graph.*, vol. 21, no. 11, pp. 1251–1258, Nov. 2015.

[36] T. Schöps, T. Sattler, C. Häne, and M. Pollefeys, "3D Modeling on the Go: Interactive 3D reconstruction of large-scale scenes on mobile devices," in *Proc. of the Int. Conf. on 3D Vis.*, Oct. 2015, pp. 291–299.

[37] R. a. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Proc. of the IEEE Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2320–2327.

[38] M. Pizzoli, C. Forster, and D. Scaramuzza, "REMODE: Probabilistic, monocular dense reconstruction in real time," in *Proc. of the IEEE Int. Conf. on Robot. and Autom.*, May 2014, pp. 2609–2616.

[39] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *Proc. of the IEEE Int. Conf. on Pattern Recognition*, Jun. 2012, pp. 3354–3361.

[40] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vis.*, vol. 47, no. 1/3, pp. 7–42, 2002.

[41] R. T. Collins, "A space-sweep approach to true multi-image matching," in *Proc. of the IEEE Int. Conf. on Pattern Recognition*, vol. 15, 1996, pp. 358–363.

[42] K. Zhang, J. Lu, and G. Lafruit, "Cross-based local stereo matching using orthogonal integral images," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 7, pp. 1073–1079, Jul. 2009.

[43] H. Heiko, "Stereo processing by semiglobal matching and mutual information," *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. 30, no. 2, pp. 328–341, Feb. 2008.

[44] M. Klingensmith, I. Dryanovski, S. S. Srinivasa, and J. Xiao, "CHISEL : Real time large scale 3D reconstruction onboard a mobile device using spatially-hashed signed distance fields," in *Proc. of Robot.: Sci. and Syst.*, Jul. 2015.