Information Fusion in Navigation Systems via Factor Graph Based Incremental Smoothing

Vadim Indelman[a], Stephen Williams[a], Michael Kaess[b], Frank Dellaert[a]

[a] *College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA*
[b] *Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA 02139, USA*

Abstract

This paper presents a new approach for high-rate information fusion in modern inertial navigation systems, that have a variety of sensors operating at different frequencies. Optimal information fusion corresponds to calculating the maximum a posteriori estimate over the joint probability distribution function (pdf) of all states, a computationally-expensive process in the general case. Our approach consists of two key components, which yields a flexible, high-rate, near-optimal inertial navigation system. First, the joint pdf is represented using a graphical model, the factor graph, that fully exploits the system sparsity and provides a plug and play capability that easily accommodates the addition and removal of measurement sources. Second, an efficient incremental inference algorithm over the factor graph is applied, whose performance approaches the solution that would be obtained by a computationally-expensive batch optimization at a fraction of the computational cost. To further aid high-rate performance, we introduce an equivalent IMU factor based on a recently developed technique for IMU pre-integration, drastically reducing the number of states that must be added to the system. The proposed approach is experimentally validated using real IMU and imagery data that was recorded by a ground vehicle, and a statistical performance study is conducted in a simulated aerial scenario. A comparison to conventional fixed-lag smoothing demonstrates that our method provides a considerably improved trade-off between computational complexity and performance.

Keywords: inertial navigation, multi-sensor fusion, graphical models, incremental inference, plug and play architecture

1. Introduction

In the past two decades, autonomous mobile robotic systems have been used in a variety of research, consumer, and military applications. Accurate and reli-

Email addresses: indelman@cc.gatech.edu (Vadim Indelman), swilliams8@gatech.edu (Stephen Williams), kaess@mit.edu (Michael Kaess), frank@cc.gatech.edu (Frank Dellaert)

able navigation is a key requirement in such systems and has been at the focus of many recent research efforts. While in early inertial navigation systems, the inertial measurement unit (IMU) was the prime sensor, modern systems have additional sensing capabilities, such as global positioning system (GPS) receivers, monocular and stereo camera systems, and range sensors. These sensors typically operate at different frequencies and are often asynchronous. Calculating a navigation solution thus becomes a matter of *information fusion*. What makes the problem even more challenging is the requirement to produce an estimate in real time.

The information fusion problem can be formulated as calculating the maximum a posteriori (MAP) estimate of the posterior probability of the system states over time, given all available measurements. The optimal solution involves performing a batch optimization each time a new measurement is received. This approach, also known as bundle adjustment (BA), is commonly used in the robotics community for solving the full simultaneous localization and mapping (SLAM) problem [21, 30, 4, 8, 6]. Recently, batch optimization has been applied for information fusion in inertial navigation systems [24, 25, 2]. However, since a batch optimization involves recalculating all the variables each time it is executed, high-rate performance quickly becomes infeasible. Thus, the systems described in [24, 25, 2] only perform batch optimizations periodically or entirely offline.

To obtain high-rate performance, standard methods use variants of the extended Kalman filter (EKF). The EKF obtains real-time performance by marginalizing out all past states and estimating only the current state. This approach has been used in a variety of applications, such as estimating the pose and the velocity of a spacecraft based on previously mapped landmarks [31], and INS in-flight-alignment [19]. While the Kalman filter provides an optimal solution in the linear case [7], most sensor models include non-linearities. The EKF, and even variants such as the iterated EKF that support re-linearization, are unable to perform a proper re-linearization since past states are no longer part of the filter.

The augmented-state EKF and fixed-lag smoothers partially overcome this problem by maintaining some of the past states within the filter. For example, an augmented-state EKF is used in [24] to incorporate information provided by multiple observations of visual features into a navigation solution, with the filter state consisting of the current navigation state and past poses. In [29], the augmented-state EKF is applied to fuse visual odometry with inertial navigation, while [28] uses a fixed-lag smoother for structure reconstruction based on stereo images that are acquired by an autonomous platform.

However, increasing the state size has a major impact on computational complexity. Current state-of-the-art techniques require updating *all* the variables in the sliding window each time *any* measurement arrives. In practice, this is not always required, since some of the state variables remain unchanged in certain conditions.

Apart from the computational complexity aspect, information fusion using fixed-lag smoothers is still sub-optimal, approaching the MAP estimate only

for sufficiently large lags. Further, as described in [12], filters and fixed-lag smoothers may become probabilistically inconsistent when the marginalized variables were linearized around an operating point that is different in the current system. This is also the case with the commonly used navigation-aiding framework [7], where IMU measurements are processed in real time into a navigation solution outside of the estimator using the most recent estimates of the navigation solution.

In this paper we present a new approach for information fusion in inertial navigation systems that addresses the deficiencies of the standard solutions. We represent the information fusion problem using a graphical model known as a *factor graph* [20]. Factor graphs encode the connectivity between the unknown variable nodes and the received measurements. Incorporating measurements from different, possibly asynchronous, sensors becomes a matter of connecting factors defined by these measurements to the appropriate nodes in the factor graph. Using factor graphs allows a plug and play capability, as new sensors are simply additional sources of factors that get added to the graph. Likewise, if a sensor becomes unavailable due to signal loss or sensor fault, the system simply refrains from adding the associated factors; no special procedure or coordination is required.

Calculating the MAP estimate is equivalent to performing inference over the factor graph. Calculating the full navigation solution over all states can be performed efficiently using a recently-developed incremental smoothing technique [16]. While this seems computationally expensive, the incremental smoothing approach exploits the system sparsity and graph topology, optimizing only a *small* portion of the nodes during each update. This is in contrast to batch optimization approaches, as well as standard fixed-lag smoothers, which always recalculate all of the state variables. Thus, the incremental smoothing approach is suitable for high frequency applications.

To further aid high-rate operation while still allowing re-linearization of the appropriate IMU measurements, we adopt a recently-developed technique of IMU pre-integration [22] and introduce the *equivalent* IMU factor that represents several consecutive IMU measurements. Thus, instead of adding variable and factor nodes to the factor graph at IMU rate, they are added at a much slower frequency that is determined by other available sensors. In contrast to navigation-aiding techniques, which also employs an IMU integrator, the IMU measurements are part of the underlying non-linear optimization in the proposed approach.

The remaining part of this paper is organized as follows. The information fusion problem is formally defined in Section 2. Section 3 then introduces the factor graph representation, while Section 4 presents factor formulations for some of the common sensors in inertial navigation systems. In particular, this section introduces the equivalent IMU factor. Information fusion via incremental smoothing is discussed in Section 5, followed by an outline of an architecture for real time performance in Section 6. Simulation and experiment results are provided in Section 7, and concluding remarks are suggested in Section 8.

2. Problem Formulation

We assume a robot is equipped with a set of multi-rate sensors, with IMU sensors typically producing measurements at high rate and sensors such as monocular or stereo cameras generating measurements at lower rates. Some sensors may become inactive from time to time (e.g. GPS), while others may be active only for short periods of time (e.g. signal of opportunity). Our goal is to calculate the best possible navigation solution by fusing all the available information sources.

More formally, let x denote the navigation state, comprising position, velocity and orientation of the robot, and denote by c any calibration parameters. In this paper we assume these represent the IMU calibration parameters (e.g. accelerometer and gyroscope biases), however any additional calibration parameters, such as camera calibration, can be included as well. Letting x_i and c_i represent, respectively, the navigation state and the calibration parameters at time instant t_i, we define the sets of all navigation states, X_k, and all calibration states, C_k, up to the current time, t_k, as

$$X_k \doteq \{x_i\}_{i=1}^{k} \quad , \quad C_k \doteq \{c_i\}_{i=1}^{k} . \tag{1}$$

In practice, due to the typical slow dynamics of the calibration variables, these variables may be introduced at a much lower frequency. In such case, C_k may be defined as $C_k \doteq \left\{c_{i_\zeta}\right\}_{\zeta=1}^{n_k}$, where $n_k \leq k$ is the number of calibration variables in C_k, and $i_\zeta \in [1, k]$.

In situations where observed landmarks are explicitly represented and estimated as part of the system state (e.g. using camera or range sensors), the jth observed landmark is denoted by l_j, and the set of all landmarks observed up to time t_k is denoted by L_k. Also, we denote by $\mathcal{V}_k$ the set of all variables up to time t_k

$$\mathcal{V}_k \doteq \{X_k, C_k, L_k\} . \tag{2}$$

Using these definitions, the joint probability distribution function (pdf) is given by

$$p\left(\mathcal{V}_k | \mathcal{Z}_k\right), \tag{3}$$

where $\mathcal{Z}_k$ represents all the measurements received up to the current time t_k. Denoting the set of measurements obtained at time t_i by Z_i, $\mathcal{Z}_k$ is defined as:

$$\mathcal{Z}_k \doteq \{Z_i\}_{i=1}^{k} . \tag{4}$$

The maximum a posteriori (MAP) estimate is given by

$$\mathcal{V}_k^* \doteq \arg\max_{\mathcal{V}_k} p\left(\mathcal{V}_k | \mathcal{Z}_k\right), \tag{5}$$

and, in the context of navigation, we are mainly interested in the optimal current navigation solution x_k^*.

3. Factor Graph Formulation in Inertial Navigation Systems

The joint pdf (3) can always be factorized in terms of *a priori* information and individual process and measurement models. Let $p(\mathcal{V}_0)$ represent all available prior information. Such a factorization can be written as[1]:

$$p(\mathcal{V}_k|\mathcal{Z}_k) = p(\mathcal{V}_0)\prod_{i=1}^{k}\left[p\left(x_i|x_{i-1},c_{i-1},z_{i-1}^{IMU}\right)p\left(c_i|c_{i-1}\right)\prod_{z_j\in Z_i\setminus z_i^{IMU}}p\left(z_j|\mathcal{V}_i^j\right)\right], \tag{6}$$

where we used $\mathcal{V}_i^j \subseteq \mathcal{V}_i$ to represent the variables involved in the general measurement model[2] $p\left(z_j|\mathcal{V}_i^j\right)$. For example, if z_j represents an observation of the landmark l acquired at vehicle state x_i, then $\mathcal{V}_i^j$ would be $\{x_i, l\}$, while in case of a GPS measurement $\mathcal{V}_i^j$ is simply $\{x_i\}$. The notation z^{IMU} is used to distinguish IMU measurements, that are described by a motion model, from other measurements.

The above factorization can be represented in a graphical model known as a factor graph [20]. A factor graph is a bipartite graph $G_k = (\mathcal{F}_k, \mathcal{V}_k, \mathcal{E}_k)$ with two types of nodes: *factor nodes* $f_i \in \mathcal{F}_k$ and *variable nodes* $v_j \in \mathcal{V}_k$. Edges $e_{ij} \in \mathcal{E}_k$ can exist only between factor nodes and variable nodes, and are present if and only if the factor f_i involves a variable v_j.

Each factor represents an individual term in the factorization (6) of $p(X_k, C_k, L_k|\mathcal{Z}_k)$, and therefore one can write

$$p(\mathcal{V}_k) \propto \prod_i f_i\left(\mathcal{V}_k^i\right), \tag{7}$$

where, as earlier, $\mathcal{V}_k^i$ represents a subset of variable nodes ($\mathcal{V}_k^i \subseteq \mathcal{V}_k$).

Each factor f_i represents an error function that should be minimized. The explicit expression of such a function depends on the specific term in the factorization (6) that is represented by the factor f_i. Denoting this error function by $err\left(\mathcal{V}_k^i, z_i\right)$, the factor f_i is defined as

$$f_i(\mathcal{V}_k^i) = d\left(err_i\left(\mathcal{V}_k^i, z_i\right)\right), \tag{8}$$

where the operator $d(.)$ denotes a certain cost function.

For Gaussian noise distributions, the general factor f_i (8) assumes the following form:

$$f_i(\mathcal{V}_k^i) = \exp\left(-\frac{1}{2}\left\|err_i\left(\mathcal{V}_k^i, z_i\right)\right\|_{\Sigma_i}^2\right), \tag{9}$$

[1] Using the definition (1) of C_k.

[2] Note that causality is enforced by the definition of $\mathcal{V}_i^j$, so that the measurement model $p\left(z_j|\mathcal{V}_i^j\right)$ for a measurement $z_j \in Z_i$ involves variables only up to time t_i.

which defines the cost function $d(.)$, and calculating the MAP estimate (5) becomes equivalent to minimizing the following non-linear least-squares function:

$$\sum_i \left\| err_i\left(\mathcal{V}_k^i, z_i\right)\right\|_{\Sigma_i}^2. \tag{10}$$

Here $\|a\|_\Sigma^2 \doteq a^T\Sigma^{-1}a$ is the squared Mahalanobis distance and Σ is the covariance matrix.

Specifically, a factor that represents a measurement model is defined as

$$f_i(\mathcal{V}_k^i) = \exp\left(-\frac{1}{2}\left\|h_i(\mathcal{V}_k^i) - z_i\right\|_{\Sigma_i}^2\right), \tag{11}$$

where $h_i(.)$ is the non-linear measurement function that predicts the sensor measurement given the variables $\mathcal{V}_k^i$, and z_i is the actual measurement. A factor that accommodates an implicit measurement function h_i is defined as

$$f_i(\mathcal{V}_k^i) = \exp\left(-\frac{1}{2}\left\|h_i(\mathcal{V}_k^{i_1}, z_i) - \mathcal{V}_k^{i_2}\right\|_{\Sigma_i}^2\right), \tag{12}$$

where $\mathcal{V}_k^{i_1}$ and $\mathcal{V}_k^{i_2}$ are two different subsets of $\mathcal{V}_k^i$ and $\mathcal{V}_k^{i_1} \cup \mathcal{V}_k^{i_2} = \mathcal{V}_k^i$. Finally, a factor that represents a motion model, that predicts the values of variables $\mathcal{V}_k^{i_2}$ based on the values of variables $\mathcal{V}_k^{i_1}$, is defined as

$$f_i(\mathcal{V}_k^i) = \exp\left(-\frac{1}{2}\left\|h_i(\mathcal{V}_k^{i_1}) - \mathcal{V}_k^{i_2}\right\|_{\Sigma_i}^2\right). \tag{13}$$

Calculating the MAP estimate (5) is equivalent to performing inference over the factor graph. While, in general, this can be an expensive operation, we show in Section 5 that using incremental smoothing, a recently developed approach [16] in the SLAM community, the involved computational complexity is *small* and high-rate performance is possible in typical navigation applications.

Before discussing the inference engine, we first present factor formulations of some of the common sensors in modern navigation systems. In particular, we introduce the *equivalent IMU factor* that accommodates chunks of consecutive IMU measurements to alleviate the necessity of adding navigation states into the optimization at IMU rate. This is essential to maintain high-rate performance.

4. Factor Formulations for Common Sensors

This section presents factor formulations for different measurement models, covering some of the common sensors in typical navigation applications. The considered sensors are IMU, GPS, and monocular and stereo cameras.

4.1. Prior Factor

The available prior information, $p(\mathcal{V}_0)$, can be factorized further into individual priors on the appropriate variables, each of which can be represented by a separate prior factor. A prior factor for some variable $v \in \mathcal{V}_0$ is a unary factor defined as

$$f^{Prior}(v) \doteq d(v). \tag{14}$$

For a Gaussian distribution, the prior information is given in terms of a mean μ_v and a covariance Σ_v, in which case the prior factor becomes $\exp\left(-\frac{1}{2}\left\|v-\mu_v\right\|^2_{\Sigma_v}\right)$. In the general case, prior information may also relate between different variables in $\mathcal{V}_0$.

4.2. IMU Factor

The time evolution of the navigation state x can be conceptually described by the following continuous non-linear differential equations (cf. Appendix A):

$$\dot{x} = h_c\left(x, c, f^b, \omega^b\right), \tag{15}$$

where f^b and ω^b are the specific force and the angular acceleration, respectively, measured by the inertial sensors in the body frame[3]. The IMU calibration parameters represented by c (cf. Section 2) are used for correcting the IMU measurement f^b, ω^b according to the assumed IMU error model. This model of IMU errors is usually estimated in conjunction with the estimation of the navigation state. Linearization of (15) will produce the well known state space representation with the appropriate Jacobian matrices and a process noise [7], which is assumed to be zero-mean Gaussian noise.

In the general case, the time propagation of c can be described according to some non-linear model of its own (e.g. random walk):

$$\dot{c} = g_c(c). \tag{16}$$

Throughout this paper, the term bias vector (or bias node) is often used when referring to the IMU calibration parameters c, although in practice this variable can represent any model of IMU errors.

A given IMU measurement $z_k^{IMU} \doteq \left\{f^b, \omega^b\right\}$ relates between the navigation states at two consecutive time instances t_k and t_{k+1}. Different numerical schemes, ranging from a simple Euler integration to high-order Runge-Kutta integration, can be applied for solving these equations. However, the factor graph framework allows the adoption of a simple Euler integration prediction function with an associated integration uncertainty. The underlying non-linear optimization will adjust individual state estimates appropriately. The discrete representation of the continuous formulation (15)-(16) is

$$\begin{aligned} x_{k+1} &= h\left(x_k, c_k, z_k^{IMU}\right) \\ c_{k+1} &= g(c_k). \end{aligned} \tag{17}$$

[3]For simplicity it is assumed that the IMU and the body frames coincide.

If desired, more sophisticated schemes can be used as well.

Conceptually, each of the equations in (17) defines a factor connecting related nodes in the factor graph: an IMU factor f^{IMU} connecting the navigation nodes x_k, x_{k+1} and the bias node c_k, and a bias factor f^{bias} connecting the bias nodes c_k and c_{k+1}. In practice, consecutive IMU measurements will be combined into a single factor, as explained in the next subsection. However, it is still useful to write down explicit expressions for the conventional IMU factor f^{IMU}.

A conventional IMU factor f^{IMU} for a given IMU measurement z_k^{IMU} is defined as follows. The IMU measurement z_k^{IMU} and the current estimate of x_k, c_k are used to predict the values for x_{k+1}. The difference between this prediction and the current estimate of x_{k+1} is the error function represented by the factor:

$$f^{IMU}\left(x_{k+1}, x_k, c_k\right) \doteq d\left(x_{k+1} - h\left(x_k, c_k, z_k\right)\right), \tag{18}$$

When adding the new variable node x_{k+1} to the graph, a reasonable initial value for x_{k+1} is required. This value can be taken, for example, from the prediction $h\left(\hat{x}_k, \hat{c}_k, z_k\right)$ where $\hat{x}_k, \hat{c}_k$ are the most recent estimates of x_k, c_k.

In a similar manner, the bias factor associated with the calculated model of IMU errors is given by

$$f^{bias}\left(c_{k+1}, c_k\right) \doteq d\left(c_{k+1} - g\left(c_k\right)\right) \tag{19}$$

with c_{k+1} and c_k represented in the factor graph as variable nodes. Note that due to the typically slow dynamics of the IMU calibration parameters c_i, the factor f^{bias} and the actual calibration nodes can be added to the factor graph at a significantly lower frequency than IMU rate [14].

Figure 1a illustrates a factor graph in a basic scenario of processing three consecutive IMU measurements. The shown factor graph contains prior, IMU and bias factors.

In practice, introducing new variables to the optimization at IMU frequency is *not* a good idea. As discussed in Section 5, high-rate performance will become infeasible in the presence of measurements of additional sensors, since the non-linear optimization will need to repeatedly re-linearize many variables.

Instead, consecutive IMU measurements can be combined into an *equivalent IMU factor*, relating between two *distant* navigation and calibration nodes, thereby avoiding introducing new variables into the optimization at IMU rate.

4.3. Equivalent IMU Factor

In this section we adopt a recently-developed technique [22] for IMU measurements pre-integration and introduce the *equivalent* IMU factor.

The idea is to integrate consecutive IMU measurements between two time instances t_i and t_j into a single component, denoted by $\Delta x_{i\to j}$, comprising the accumulated change in position, velocity and orientation. Thus, $\Delta x_{i\to j}$ relates between the navigation states at the two time instances, x_i and x_j, and the calibration parameters c_i that are used for correcting the IMU measurements. Following [22], we refer to $\Delta x_{i\to j}$ as the *pre-integrated delta* and summarize the equations for its calculation in Appendix B.

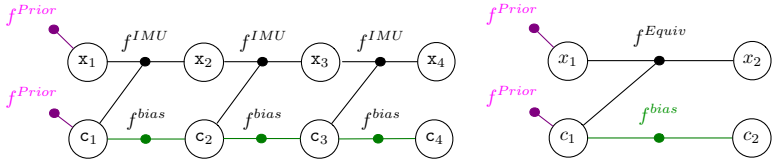


Figure 1: (a) Factor graph representation in a basic scenario of three IMU measurements. Navigation and IMU calibration nodes x_i and c_i are added to the factor graph at IMU rate. Given t_1 and IMU frequency $1/\Delta t$, node indices correspond to the time instances $t_1, t_1 + \Delta t, t_1 + 2\Delta t$ and $t_1 + 3\Delta t$. (b) Factor graph representation using an equivalent IMU factor, which accommodates all the consecutive IMU measurements between t_1 and *some* t_2. Navigation and calibration nodes are introduced only when a new factor f^{Equiv} is added, at a much lower rate than IMU rate. Different notations are used in (a) and (b) to distinguish between the IMU-rate nodes and the nodes added using an equivalent IMU factor ($\mathtt{x}_i, \mathtt{c}_i$ and x_i, c_i).

A high-rate navigation solution, x_j, can be calculated based on $\Delta x_{i\to j}$ and the current estimates of x_i and c_i. Additionally, at *chosen* time instances, the pre-integrated delta, $\Delta x_{i\to j}$, can be used within an equivalent IMU factor f^{Equiv}, capturing the error between the predicted and actual navigation state at t_j:

$$f^{Equiv}\left(x_j, x_i, c_i\right) \doteq d\left(x_j - h^{Equiv}\left(x_i, c_i, \Delta x_{i\to j}\right)\right), \tag{20}$$

where h^{Equiv} represents the non-linear function that predicts x_j given x_i, c_i and $\Delta x_{i\to j}$. An explicit expression for h^{Equiv} is detailed in Appendix B.

The factor f^{Equiv} represents all the IMU measurements between t_i and t_j, yet incorporating it into the factor graph involves adding *only* the variable x_j. This is in contrast to adding variables at IMU rate in the case of a conventional IMU factor.

Figure 1b illustrates a factor graph with a single equivalent IMU factor and a bias factor for some two time instances $t_i = t_1$ and $t_j = t_2$. When $t_2 = t_1 + 3\Delta t$ with $1/\Delta t$ denoting the IMU frequency, the two factor graphs in Figures 1a-1b accommodate the same number of IMU measurements.

Since the non-linear cost function (5) is optimized by repeated linearization (cf. Section 5), the computational cost of re-linearizing each factor is of prime importance. A straightforward approach for calculating $\Delta x_{i\to j}$ involves integrating the IMU measurements while expressing them in the navigation frame. However, linearizing f^{Equiv} in such approach requires *recalculating* $\Delta x_{i\to j}$ from scratch whenever the rotation estimate changes. Clearly, this is an expensive operation that should be avoided.

To resolve this issue, [22] proposed to express the different components in $\Delta x_{i\to j}$ in the *body* frame of the first time instant (i.e. t_i), instead of the nav-

igation frame. The predicting function h^{Equiv} is adjusted accordingly so that $h^{Equiv}(x_i, c_i, \Delta x_{i\to j})$ predicts x_j. Consequently, $\Delta x_{i\to j}$ does not depend on any particular values of x_i and x_j, and therefore the equivalent factor can be re-linearized *without* recalculating $\Delta x_{i\to j}$.

Remark 1 As mentioned, calibration parameters do not change significantly over time and therefore it makes sense to introduce calibration nodes only when an equivalent IMU factor is added, or at some lower frequency. The factor f^{Equiv} depends only on c_i (and not c_j) due to the simple Euler integration scheme that was also applied in the case of a conventional IMU factor (cf. Section 4.2).

Remark 2 At this point it is useful to state the approximations involved with the equivalent IMU factor. These involve assuming that the gravity vector and the rotational rate of the navigation frame with respect to an inertial navigation system are constant between t_i and t_j (cf. Appendix B). In addition, in order to avoid re-calculating $\Delta x_{i\to j}$, the effect of re-linearization of calibration parameters c_i on $\Delta x_{i\to j}$ is accounted by considering a first order approximation. See [22] for further details.

4.4. GPS Measurements

The GPS measurement equation is given by

$$z_k^{GPS} = h^{GPS}(x_k) + n^{GPS}, \tag{21}$$

where n^{GPS} is the measurement noise and h^{GPS} is the measurement function, relating between the measurement z_k^{GPS} to the robot's position. In the presence of lever-arm, rotation will be part of the measurement equation as well [7]. The above equation defines a unary factor

$$f^{GPS}(x_k) \doteq d\left(z_k^{GPS} - h^{GPS}(x_k)\right), \tag{22}$$

which is only connected to the node x_k that represents the navigation state at time t_k. The GPS factor f^{GPS} can be added to the graph, in conjunction with the equivalent IMU factor f^{Equiv} that uses the current pre-integrated delta $\Delta x_{i\to k}$ (here i denotes the most recent navigation node in the graph).

Examples of factor graphs with GPS measurements and measurements from other sensors, operating at different rates, are shown in Figures 2a-2c. Prior factors on navigation and calibration nodes are shown as well. GPS pseudo-range measurements can be accommodated in a similar manner.

4.5. Monocular and Stereo Vision Measurements

Incorporating vision sensors can be done on several levels, depending on the actual measurement equations and the assumed setup.

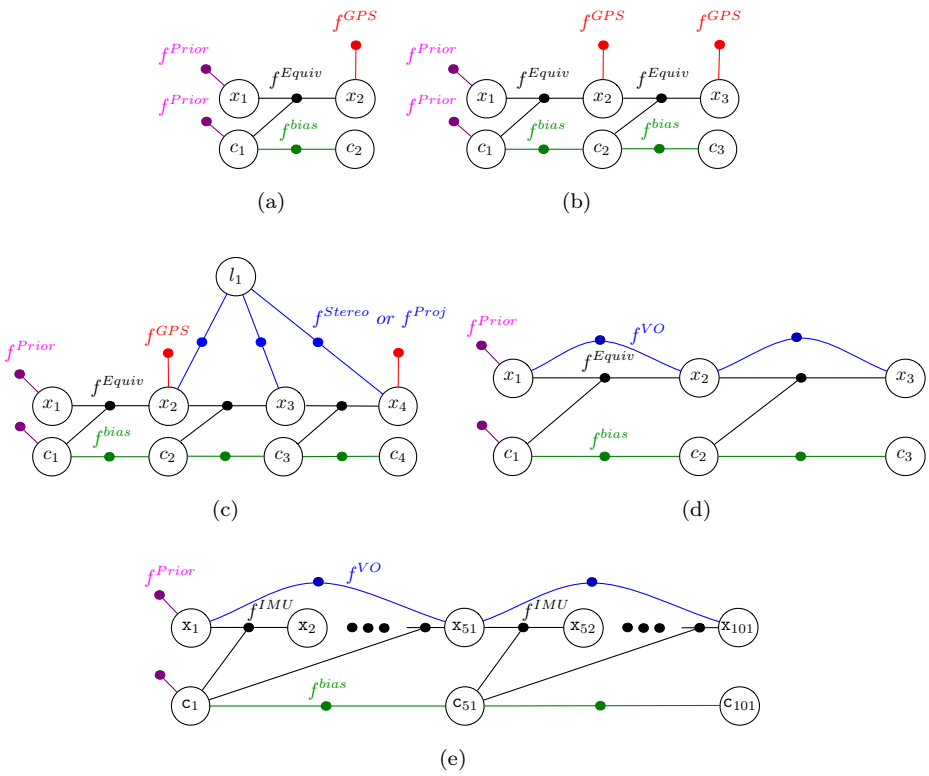


Figure 2: (a) and (b): Factor graphs that involve GPS, equivalent IMU, bias and prior factors. (c) Factor graph that also accommodates factors for monocular or stereo camera observations. (d) Factor graph with equivalent IMU, visual odometry, bias and prior factors. (e) An equivalent factor graph to (d) when using conventional IMU factors, assuming visual odometry measurements are obtained every 50 IMU measurements and bias nodes are added at the same frequency as in (d). The factor graph (e) contains many more variable nodes and thus optimization will be substantially slower than when using an equivalent IMU factor (cf. Section 5.2). Different notations are used in (e) to distinguish the IMU-rate nodes from the nodes added using an equivalent IMU factor ($\mathtt{x}_i, \mathtt{c}_i$ and x_i, c_i).

4.5.1. Monocular Camera

Assuming a monocular pinhole camera model [10] with a known calibration matrix K, an image observation is predicted by the non-linear function π defined as

$$\pi(x, l) = K \begin{bmatrix} R & t \end{bmatrix} l, \tag{23}$$

where l represents the coordinates of the observed landmark and x is the navigation state. Both the landmark coordinates and the predicted image observation are given in homogeneous coordinates. If the landmark is expressed in the global system, the rotation matrix R and the translation vector t represent the transformation between the camera and the global frame. Therefore, they can be calculated from the current estimate of the navigation state x assuming the transformation between the body and camera frame is known.

When observing known landmarks, Eq. (23) defines a unary factor on the node x. The more challenging problem of observing unknown landmarks, also known as full SLAM or BA, requires adding the unknown landmarks into the optimization by including them as variable nodes in the factor graph. A projection factor is then defined as

$$f^{Proj}(x, l) \doteq d\left(z - \pi(x, l)\right), \tag{24}$$

where z represents the actual image observation.

Alternatively, to avoid including the unknown landmarks into the optimization, one can use multi-view constraints [10, 23], such as two-view [5] and three-view constraints [13], instead of the projection equation (23).

4.5.2. Stereo Camera

A stereo camera rig is another commonly used setup. Assuming a known baseline, one can incorporate the observed landmark l as a variable into the optimization and add a factor that represents the projection of the observed landmark onto the image plane of the two cameras. Such a factor is defined as

$$f^{Stereo}(x, l) \doteq d\left(z^R - \pi^R(x, l)\right) d\left(z^L - \pi^L(x, l)\right), \tag{25}$$

where the superscripts L and R represent the left and right cameras, respectively.

Alternatively, it is possible to first estimate the relative transformation, T_Δ, between two different stereo frames at time instances t_i and t_j using all of the landmark observations in those frames. Such a transformation can be used to predict the robot pose at t_j based on x_i, with the corresponding visual-odometry binary factor $f^{VO}(x_i, x_j)$ [14].

Figure 2c illustrates a factor graph in a basic multi-rate scenario. The factor graph includes either projection (24) or stereo (25) factors, as well as GPS, equivalent IMU, bias and prior factors.

5. Information Fusion via Incremental Smoothing

So far, the information fusion problem was formulated using a factor graph representation and factors for some common sensors were defined. This section discusses incremental smoothing, an inference algorithm over factor graphs

that produces nearly optimal MAP estimates of (3), with low computational complexity.

We start with describing a conventional batch optimization and then proceed to incremental smoothing. In Section 5.3 we also analyze the relation to fixed-lag smoothing, EKF and the general navigation-aiding framework.

5.1. Batch Optimization

We solve the non-linear optimization problem encoded by the factor graph by repeated linearization within a standard Gauss-Newton style non-linear optimizer. Starting from an initial estimate $\hat{\mathcal{V}}_k$ of the set of all variables $\mathcal{V}_k$, Gauss-Newton finds an update Δ from the linearized system

$$\arg\min_{\Delta} \|J(\hat{\mathcal{V}}_k)\Delta - b(\hat{\mathcal{V}}_k)\|^2, \tag{26}$$

where $J(\hat{\mathcal{V}}_k)$ is the sparse Jacobian matrix at the current linearization point $\hat{\mathcal{V}}_k$ and $b(\hat{\mathcal{V}}_k)$ is the residual given all measurements thus far $\mathcal{Z}_k$ (cf. Section 2). The Jacobian matrix is equivalent to a linearized version of the factor graph, and its block structure reflects the structure of the factor graph. After solving equation (26), the linearization point is updated to the new estimate $\hat{\mathcal{V}}_k + \Delta$.

Solving for the update Δ requires factoring the Jacobian matrix into an equivalent upper triangular form using techniques such as QR or Cholesky. Within the factor graph framework, these same calculations are performed using variable elimination [11]. A variable ordering is selected and each node is sequentially eliminated from the graph, forming a node in a chordal Bayes net [27]. A Bayes net is a directed, acyclic graph that encodes the conditional densities of each variable. Chordal means that any undirected loop longer than three nodes has a chord or a short cut.

The Bayes net is equivalent to the upper triangular matrix R that results from the Jacobian matrix factorization (e.g. $J = QR$), and thus represents the square root information matrix. Given a Bayes net, the update Δ is obtained by back-substitution.

Each node in the Bayes net represents a conditional probability. When a variable node v is eliminated from the factor graph, all the factors $f_i\left(\mathcal{V}^i\right)$ involving that node (i.e. $v \in \mathcal{V}^i$) are identified and re-factorized as

$$\prod_i f_i\left(\mathcal{V}^i\right) = p\left(v|S\right) f\left(S\right), \tag{27}$$

where S denotes the set of variables involved in the factors $f_i\left(\mathcal{V}^i\right)$, excluding v: $S = \left\{v_j | \exists i : \ v_j \in \mathcal{V}^i \ , \ v_j \neq v\right\}$. The conditional $p\left(v|S\right)$ is then added to the Bayes net, while the new factor $f\left(S\right)$ is added to the factor graph. The Bayes net graphical model expresses the conditioning relations between the different variable nodes with directed edges (see illustration in Figures 3a-3b and an example below). Further details can be found in [15].

While the elimination order is arbitrary and any order will form an equivalent Bayes net, the selection of the elimination order does affect the *structure*

of the Bayes net and the corresponding amount of computation. A good elimination order will make use of the natural sparsity of the system to produce a small number of edges, while a bad order can yield a fully connected Bayes net. Heuristics do exist, such as approximate minimum degree [3], that approach the optimal ordering for generic problems.

Example As a basic example, consider the factor graph given in Figure 2a, which was already discussed in Section 4.5. The corresponding Jacobian matrix J and the upper triangular matrix R, obtained from a QR factorization $J = QR$, are of the following form:

$$J = \begin{array}{c} \begin{matrix} x_1 & c_1 & x_2 & c_2 \end{matrix} \\ \begin{bmatrix} \times & & & \\ & \times & & \\ \times & \times & \times & \\ & \times & & \times \\ & & \times & \end{bmatrix} \end{array} \quad , \quad R = \begin{array}{c} \begin{matrix} x_1 & c_1 & x_2 & c_2 \end{matrix} \\ \begin{bmatrix} \times & \times & \times & \\ & \times & \times & \times \\ & & \times & \times \\ & & & \times \end{bmatrix} \end{array} \tag{28}$$

Non-zero entries are denoted by $\times$. Each row in J represents a single linearized factor. For example, the third row represents the linearized equivalent IMU factor and therefore it involves the variables: x_1, c_1 and x_2. The matrix R represents the square root information matrix.

Sequential elimination of the variables from the factor graph, assuming the variable elimination ordering x_1, c_1, x_2, c_2, leads to the Bayes net shown in Figure 3a. The Bayes net represents the matrix R: each node represents a conditional probability, as detailed in the figure, and the overall graphical structure encodes a factorization of the joint pdf $p(x_1, x_2, c_1, c_2)$:

$$p(x_1, x_2, c_1, c_2) = p(c_2)\, p(x_2|c_2)\, p(c_1|x_2, c_2)\, p(x_1|c_1, x_2)\,. \tag{29}$$

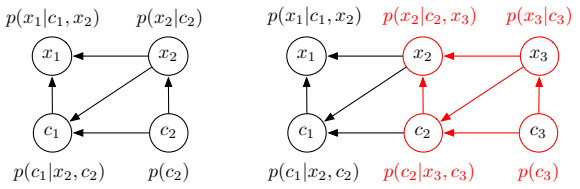
The arrows in Figure 3a express the above factorization. For example, the conditional $p(c_1|x_2, c_2)$ is represented by the two arrows: from x_2 to c_1 and from c_2 to c_1.

5.2. Incremental Smoothing

In the context of the navigation smoothing problem, each new sensor measurement will generate a new factor in the graph. This is equivalent to adding a new row (or block-row in the case of multi-dimensional states) to the measurement Jacobian of the linearized least-squares problem.

The key insight is that optimization can proceed incrementally because most of the calculations are the same as in the previous step and can be reused. When using the algorithm of Section 5.1 to recalculate the Bayes net, one can observe that only part of the Bayes net is modified by the new factor.

Therefore, instead of recalculating the Bayes net from scratch, we focus computation on the affected part of the Bayes net. Informally, once the affected part of the Bayes net has been identified, it is converted back into a factor



(a) (b)

Figure 3: (a) Bayes net that corresponds to eliminating the factor graph shown in Figure 2a using elimination order x_1, c_1, x_2, c_2. (b) Adding new GPS and accumulated IMU measurements results in a factor graph that is shown in Figure 2b, with new nodes x_3 and c_3. Incremental smoothing allows processing only part of the Bayes net, indicated in red color, instead of re-calculating it from scratch.

graph, the new factor is added, and the resulting (small) factor graph is re-eliminated. The eliminated Bayes net is then merged with the unchanged parts of the original Bayes net, creating the same result as a batch solution would obtain. A formal exposition is given in the incremental smoothing algorithm iSAM2 [16] that we apply here.

The affected part of the Bayes net, that we denote by aBN, can be identified following Algorithm 1 (see also example below). In practice, the incremental smoothing algorithm identifies aBN more efficiently (the reader is referred to [16] for further details).

Algorithm 1 Identification of affected parts in the Bayes net

1: **Input**: Bayes net, new factors
2: **Initialization**: $aBN = \phi$
3: Locate all the variable nodes that are involved in the new factors
4: **for** each such node v **do**
5: Add v to aBN
6: Locate all paths in the Bayes net that lead from the last eliminated node to the node v
7: **for** each such path **do**
8: **if** a node v' is on the path and $v' \notin aBN$ **then**
9: Add v' to aBN
10: **end if**
11: **end for**
12: **end for**

To deal with non-linear problems efficiently, iSAM2 combines the incremental updates with selective re-linearization by monitoring the validity of the linearization point of each variable. A variable is marked for re-linearization if

its Δ from the previous factorization is above a specified threshold. Different thresholds are used for each variable type.

As long as only sequential IMU measurements are processed, regardless of whether the conventional or equivalent IMU factors are used, the resulting factor graph and the Bayes net will have a chain-like structure (cf. Figure 1). Only a small part of the Bayes net is modified each time a new IMU factor is added. For the examples given in Figure 1, the Bayes net can be updated, i.e. re-factoring the Jacobian into a square root information matrix, by modifying only 4 of its nodes, regardless of the actual size of the graph [14].

The effectiveness of the equivalent IMU factor over the conventional IMU treatment becomes apparent when additional sensors are available. The affected part of the Bayes net that must be recomputed is far larger when using a conventional IMU factor, since the navigation (and calibration) variables are added at IMU frequency. In contrast, the equivalent IMU factor only requires new variables to be added to the optimization (i.e. adding new variable nodes into the factor graph) when a measurement from an additional sensor is received. As a result, fewer variables in the Bayes net must be recalculated, leading to a significant improvement in computational complexity.

Figures 2d and 2e show an example of factor graphs when using visual odometry factors f^{VO} and either conventional or equivalent IMU factors. Assuming visual odometry measurements are obtained every 50 IMU measurements, the two factor graphs represent the same amount of information. However, updating the Bayes net with a new visual odometry factor involves modifying approximately 50 nodes when using a conventional IMU factor and only 4 nodes in the case of the equivalent IMU factor.

Example To illustrate that only part of the Bayes net, and equivalently the square root information matrix, changes when new measurements are incorporated, we continue the example from the previous section and assume that a new GPS measurement is received at time t_3. To incorporate this measurement into the factor graph (i.e. into the optimization), new navigation and IMU calibration nodes x_3 and c_3 are introduced, as well as GPS, bias and equivalent IMU factors. The equivalent IMU factor, in this case, accommodates all the IMU measurements between the time of the previous GPS measurement, t_2, and t_3. The new factor graph is shown in Figure 2b, with the new added factors $f^{Equiv}(x_3, x_2, c_2)$, $f^{bias}(c_2, c_3)$ and $f^{GPS}(x_3)$.

The updated Jacobian J', evaluated about the same linearization point as in the previous case, and its factorization into an upper triangular matrix R'

are

$$J' = \begin{array}{c} \begin{array}{cccccc} x_1 & c_1 & x_2 & c_2 & x_3 & c_3 \end{array} \\ \left[\begin{array}{cccccc} \times & & & & & \\ & \times & & & & \\ \times & \times & \times & & & \\ & \times & & \times & & \\ & & \times & & & \\ & & \underline{\times} & \underline{\times} & \underline{\times} & \\ & & & \underline{\times} & & \underline{\times} \\ & & & & \underline{\times} & \end{array}\right] \end{array}, \quad R' = \begin{array}{c} \begin{array}{cccccc} x_1 & c_1 & x_2 & c_2 & x_3 & c_3 \end{array} \\ \left[\begin{array}{cccccc} \times & \times & \times & & & \\ & \times & \times & \times & & \\ & & \underline{\times} & \underline{\times} & \underline{\times} & \\ & & & \underline{\times} & \underline{\times} & \underline{\times} \\ & & & & \underline{\times} & \underline{\times} \\ & & & & & \underline{\times} \end{array}\right] \end{array},$$

where the *new or modified* non-zero entries are marked in bold and an underline. As seen, the Jacobian has three new rows that correspond to the new factors, and two new variables x_3 and c_3. Referring to the matrix R', the first two block-rows remain unchanged, and therefore there is no need to recalculate these entries. The corresponding Bayes net is given in Figure 3b, with the *modified* conditionals denoted in red.

Identifying the affected part in the Bayes net from the previous step (Figure 3a), can be done as explained in Section 5.2: The new factors involve existing variable nodes x_2 and c_2. The node c_2 is also the last-eliminated node. The paths from the last-eliminated node to the nodes x_2 and c_2 include only the two nodes x_2, c_2. Therefore, the affected parts in the previous Bayes net are only the conditionals $p(c_2)$ and $\mathrm{p}(x_2|c_2)$, represented by the nodes x_2 and c_2. Calculating the new Bayes net will involve modifying these conditionals and adding new conditionals for the new variable nodes x_3 and c_3. The conditionals $p(x_1|c_1, x_2)$ and $p(c_1|x_2, c_2)$ remain unchanged.

5.3. Relation to Fixed-Lag Smoother, EKF and Navigation-Aiding

In this section, we discuss the conceptual differences of incremental smoothing with fixed-lag smoothing, EKF and the navigation aiding framework. The following two aspects are analyzed: computational complexity and accuracy.

5.3.1. Fixed-Lag Smoother

To maintain a constant and affordable computational cost, fixed-lag smoother approaches marginalize out variables outside of the smoothing lag. *All* the variables within the smoothing lag are re-calculated each time a new measurement is obtained, which corresponds to performing a batch optimization over the variables within the smoothing lag. At each iteration, the Jacobian of the linearized fixed-lag system is re-factorized from *scratch* into an upper triangular matrix. The latter is then used for calculating the Δ by back-substitution, followed by updating the linearization point (cf. Section 5.1).

In contrast, incremental smoothing re-uses calculations by incrementally updating the appropriate part of the Bayes net, representing the square root information matrix of the *entire* system (i.e. factorization of the Jacobian matrix without marginalizing any variables). Thus, each time a new measurement

(factor) is received, incremental smoothing exploits sparsity and identifies what specific variables should be re-calculated, rather than recalculating all variables within some sliding window.

In terms of performance, fixed-lag smoother based approaches are expected to produce inferior results to incremental smoothing when the smoothing lag is not large enough to accommodate all the variables that are updated by an incoming measurement.

Marginalizing out a variable involves discarding any measurements (factors) that involve that variable (node) and representing them as a single Gaussian evaluated at the time of marginalization. This Gaussian is a *linear* factor that cannot be re-linearized as we no longer have access to the marginalized variable. Since the linear factor cannot be re-linearized, the result of the non-linear optimization will be sub-optimal and, without proper treatment of the linear components, may become inconsistent [12].

Incremental smoothing, on the other hand, uses the non-linear factor graph, which captures all the available information to perform inference. Actual performance depends on the specific values for linearization thresholds, which act as tuning parameters. Our observation is that setting these thresholds to reasonable values for each variable type produces results that approach the optimal MAP estimation produced by a batch optimization, as demonstrated in Section 7.

5.3.2. EKF

The relation of incremental smoothing to different variations of the well-known EKF can be inferred from the tight connection of the latter to a fixed-lag smoother. In particular, a fixed-lag smoother is equivalent to an iterated augmented-state EKF when the state vector includes all the variables within the smoothing lag and both the prediction and update steps are iterated. The more basic version, the augmented-state EKF without iterations, is expected to produce inferior results since it only involves a single linearization. Finally, a simple EKF marginalizes out all past variables, which corresponds to a fixed-lag smoother with zero lag.

5.3.3. Relation to Navigation-Aiding Techniques

Navigation-aiding techniques often separate the navigation information fusion problem into two processes: i) A high-rate process in which incoming IMU measurements are integrated into the navigation solution using the previous estimate of the navigation solution, and ii) A lower-rate process in which an error-state filter is used for calculating corrections to the navigation solution based on measurements from additional sensors. While such an architecture produces a navigation solution in real time and is capable of incorporating different asynchronous sensors, the information fusion process (and therefore the navigation solution) is sub-optimal, regardless to the actual estimator being used: Since IMU measurements are incorporated *outside* of the estimator, the non-linear factors that represent these measurements cannot be re-linearized during the estimation process. As a consequence, the optimization will fail

in achieving MAP estimates. This observation is particularly important when using low-quality IMU sensors.

6. Architecture for Real Time Performance

Although the computational complexity of incremental smoothing is significantly lower than that of a batch optimization, the actual processing time of incorporating a new measurement depends on the size of the Bayes net that needs to be recalculated (cf. Section 5.2). However, in practice, a navigation solution is required in real time. This is already possible when processing only IMU measurements: these are accumulated in the IMU pre-integrated component $\Delta x_{i\to j}$, which can be used to generate a navigation solution in real time via the predicting function $h^{Equiv}\left(\hat{x}_i, \hat{c}_i, \Delta x_{i\to j}\right)$. Yet, when measurements from additional sensors are obtained, processing time, while still being small, does not guarantee real time performance. For example, in the scenarios considered in the results section, processing time typically fluctuates between $5-45$ mili-seconds, with a few instances around 200 mili-seconds (cf. Figure 5).

Real time performance is possible, however, by *parallelization*. A high-priority process is used to pre-integrate each incoming IMU measurement at time t_j into $\Delta x_{i\to j}$. Based on the most recent estimates $\hat{x}_i, \hat{c}_i$ of x_i, c_i, a navigation solution is calculated in real time using $h^{Equiv}\left(\hat{x}_i, \hat{c}_i, \Delta x_{i\to j}\right)$. When a non-IMU measurement becomes available at some time instant t_j, incremental smoothing can be performed in a lower-priority process. This involves generating a factor representing the appropriate measurement model, as well as creating equivalent IMU and bias factors. These factors, and the new navigation and calibration nodes x_j, c_j, are added to the factor graph. The estimates of these variables (i.e. $\hat{x}_j, \hat{c}_j$) are initialized by appropriate predicting functions. In particular, x_j is predicted by $h^{Equiv}\left(\hat{x}_i, \hat{c}_i, \Delta x_{i\to j}\right)$, yielding $\hat{x}_j$. Incremental smoothing is engaged and t_j is marked as the new starting pre-integration time, followed by an initialization of the IMU pre-integration component $\Delta x_{j\to j}$ (cf. Appendix B). Navigation solution continues being calculated in real time based on $\hat{x}_j$ and $\Delta x_{j\to k}$, with index k denoting current time. Once incremental smoothing is complete, the updated estimates of x_j and c_j replace $\hat{x}_j$ and $\hat{c}_j$, without the need to modify $\Delta x_{j\to k}$.

This architecture is summarized in Algorithm 2. Note that the algorithm also supports cases in which new non-IMU measurements are obtained before incremental smoothing has finished its operation.

7. Results

In this section the presented method is evaluated both in a statistical study, using a simulated environment, and in an experiment. An aerial scenario is considered in the former, while the experiment data was collected by a ground vehicle. The method is compared, in terms of estimation accuracy and computational complexity, to a batch estimator and to a fixed-lag smoother with

Algorithm 2 Architecture for real time performance

Initialization: New factors $\mathcal{F}_{new} \doteq \{\}$, New variables $\mathcal{V}_{new} \doteq \{\}$
Initialization: Set $\hat{x}_j, \hat{c}_j$ according to available priors $p(x_0)$ and $p(c_0)$, Initialize $\Delta x_{i\to j}$
while measurement z **do**
  if z is an IMU measurement **then**
    Update $\Delta x_{i\to j}$ with z by running Algorithm 3
    Produce a navigation solution as $h^{Equiv}(\hat{x}_i, \hat{c}_i, \Delta x_{i\to j})$
  else
    Calculate predictions $\hat{x}_j, \hat{c}_j$ of x_j, c_j based on $\hat{x}_i, \hat{c}_i$ and $\Delta x_{i\to j}$
    Add x_j, c_j to $\mathcal{V}_{new}$
    Create an appropriate factor f^z for the measurement z
    Create an equivalent IMU factor $f^{Equiv}(x_i, c_i, x_j)$ and a bias factor $f^{bias}(c_i, c_j)$
    Add the factors f^z, f^{Equiv}, f^{bias} to $\mathcal{F}_{new}$
    if incremental smoothing process is available **then**
      Engage incremental smoothing algorithm (cf. Section 5.2) with the new factor and variables nodes $\mathcal{F}_{new}$ and $\mathcal{V}_{new}$
      Set $\mathcal{F}_{new} = \{\}$, and $\mathcal{V}_{new} = \{\}$
    end if
    Initialize $\Delta x_{i\to j}$
    Set $\hat{x}_i = \hat{x}_j$ and $\hat{c}_i = \hat{c}_j$
  end if
  if incremental smoothing is done **then**
    Retrieve updated estimates $\hat{x}, \hat{c}$ of most recent navigation and calibration nodes.
    Set $\hat{x}_i = \hat{x}$ and $\hat{c}_i = \hat{c}$
  end if
end while

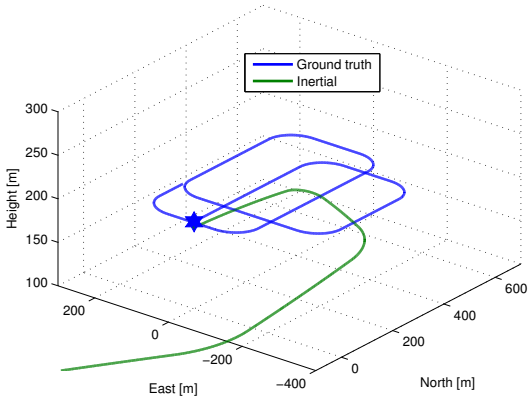


Figure 4: Ground truth trajectory used in statistical study. Inertial navigation solution for the first few seconds is shown as well. Beginning of trajectory is denoted by a mark.

different lag sizes. All methods were implemented within the GTSAM[4] optimization library and were run on a single core of an Intel i7-2600 processor with a 3.40GHz clock rate and 16GB of RAM memory.

7.1. Simulation Results

The proposed method was examined in a Monte-Carlo simulation of an aerial vehicle. A ground truth trajectory was created, simulating a flight of an aerial vehicle at a 20 m/s velocity and a constant height of 200 meter above mean ground level. The trajectory consists of several segments of straight and level flight and maneuvers, as shown in Figure 4.

Based on the ground truth trajectory, ideal IMU measurements were generated at 100 Hz, while taking into account Earth's rotation and changes in the gravity vector (cf. Appendix A). For each of the 100 Monte-Carlo runs, these measurements were corrupted with a constant bias and zero-mean Gaussian noise in each axis. Bias terms were drawn from a zero-mean Gaussian distribution with a standard deviation of $\sigma = 10$ mg for the accelerometers and $\sigma = 10$ deg/hr for the gyroscopes. The noise terms were drawn from a zero-mean Gaussian distribution with $\sigma = 100\,\mu g/\sqrt{Hz}$ and $\sigma = 0.001\,deg/\sqrt{hr}$ for the accelerometers and gyroscopes. Initial navigation errors were drawn from zero-mean Gaussian distributions with $\sigma = (10, 10, 15)$ meters for position (expressed in a north-east-down system), $\sigma = (0.5, 0.5, 0.5)$ m/s for velocity and $\sigma = (1, 1, 1)$ degrees for orientation.

[4] http://tinyurl.com/gtsam.

In addition to IMU, the aerial robot was assumed to be equipped with a monocular camera and a 3-axes magnetometer, operating at 0.5Hz and 2Hz, respectively. Ideal visual observations were calculated by projecting *unknown short-track* landmarks, scattered on the ground with ± 50 meters elevation, onto the camera. A zero-mean Gaussian noise, with $\sigma = 0.5$ pixels, was added to all visual measurements. Landmarks were observed on average by 5 views, with the shortest and longest landmark-track being 2 and 12, respectively. Loop closure measurements (i.e. landmark re-observations) were not included. Magnetometer measurements were created by contaminating the ground truth angles with a $\sigma = 3$ degrees noise in each axis.

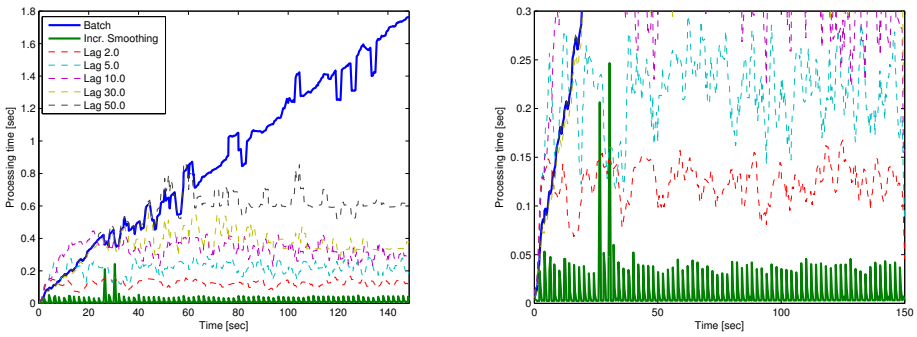
Incoming IMU measurements were accumulated and incorporated into a factor graph using an equivalent IMU factor (20), each time a measurement from the camera or the magnetometer arrived. Projection and magnetometer factors were used to incorporate the latter into the optimization. The projection factor is defined by Eq. (24), while the magnetometer factor is a simple unary factor that involves only the orientation variable, and can be defined in a similar manner as the GPS factor (22). No measurement delays were assumed. A random walk process was used for the bias factor (19), with new IMU calibration nodes added to the factor graph each time an equivalent IMU factor was added.

Figures 5-8 compare the performance and the computational cost of the proposed method to a batch optimization and to a fixed-lag smoother with $2, 5, 10, 30$ and 50 second lag size. Landmarks are kept within the smoothing lag as long as they are observed by at least one of the robot's poses in the lag, and are marginalized out when this is no longer the case.

Performance of each method is shown in Figure 6 in terms of the root mean squared error (RMSE) for each component of navigation state: position, velocity, orientation, accelerometer and gyroscope biases. In addition, Figure 7 shows the root mean squared difference between the different methods (incremental smoothing and fixed-lag smoothers) and the MAP estimate that is obtained by a batch optimization. Figure 8 provides further comparison between the proposed approach and a batch optimization, including estimated covariances.

As seen, incremental smoothing yields very similar results, in all states, to those obtained by a batch optimization (i.e. MAP estimate), while the computational cost is very small, typically within the $5-45$ milli-seconds range (cf. zoom in Figure 5b). Lower timing values ($\sim$ 5 ms) refer to the cost of incorporating magnetometer factors, while higher values ($\sim$ 45 ms) are related to adding also camera observations into the optimization (equivalent IMU and bias factors are added in both cases). The two spikes around $t = 26$ and $t = 30$ seconds correspond to the first maneuver phase. During this phase, due to increased observability, additional variables are estimated, including accelerometer bias in x and y axes, as well as roll and pitch angles. The underlying optimization (i.e. smoothing) involves re-linearization of many past variables, whose estimation is improved as well, resulting in increased computational complexity. Note that real time performance can be obtained following the architecture outlined in Section 6.

Referring to fixed-lag smoothers one can observe the trade-off between lag



(a) (b)

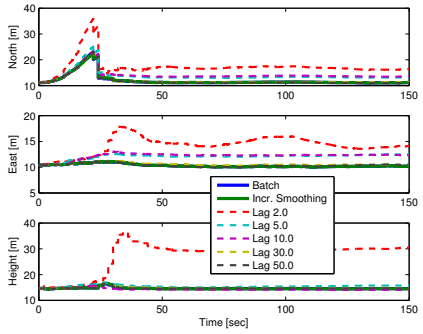
Figure 5: (a) Computational cost in a typical run. (b) Zoom in. For clarity, the presented processing time of all methods, except of incremental smoothing, has been smoothed. An original processing time of incremental smoothing is presented.

size and the computational cost: increasing the smoothing lag size leads to improved performance at the cost of higher processing time (cf. Section 5.3): on a statistical level, the difference between the MAP estimate and the solution obtained by a fixed lag smoother decreases when increasing the smoothing lag (cf. 7). As expected, the solution is identical to the MAP estimate as long as the lag is not full, while afterwards the solution becomes sub-optimal. In particular, a 2-second lag produces considerably worse results than incremental smoothing, while still being more computationally expensive. When using a 50-second lag, performance is nearly identical to the MAP estimate obtained by a batch optimization, as well to incremental smoothing, however the computational cost is very high (cf. Figures 5, 6b and 7).

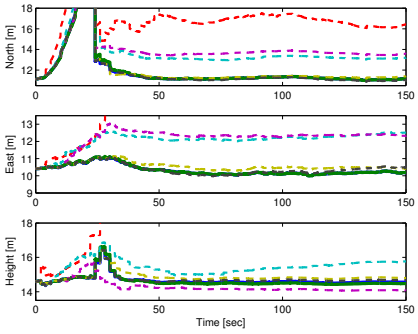
7.2. Experiment Results

To test the proposed method on real-world data, we make use of the KITTI Vision Benchmark Suite [9]. These datasets were captured from the autonomous vehicle platform "Annieway" during traverses around the city of Karlsruhe, Germany. This platform consists of a car chassis outfitted with a stereo camera and a differential GPS/INS system. The differential GPS/INS data provides highly accurate ground truth position and orientation data. Additionally, raw IMU measurements are provided at 100 Hz, and raw camera images from the stereo rig are available at 10Hz.

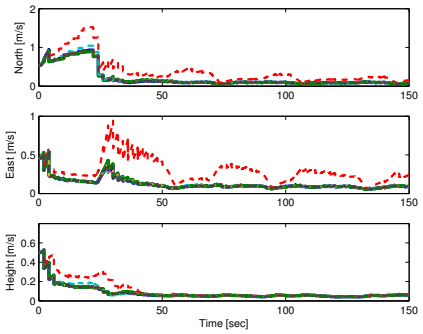
As in the simulated scenario, incoming IMU measurements were accumulated using an equivalent IMU factor (20), and only incorporated into the factor graph when a stereo image pair was processed. A random walk process was used for the bias factor (19), with new IMU calibration nodes added to the factor graph each time an equivalent IMU factor was added. To process the raw stereo images, a



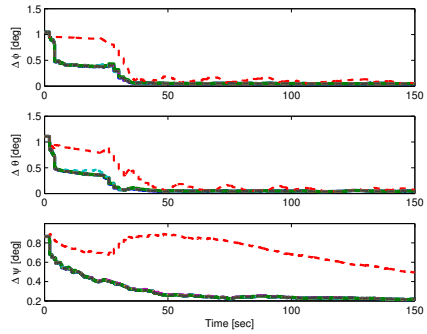
(a) Position



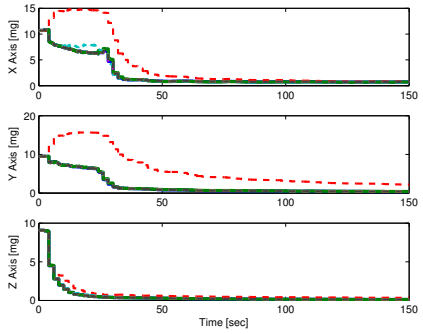
(b) Position - zoom



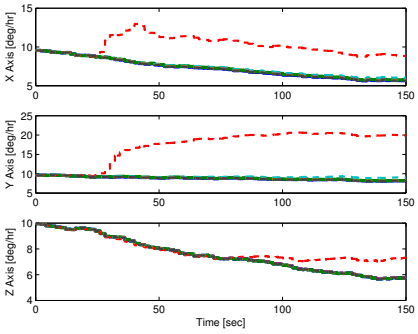
(c) Velocity



(d) Orientation

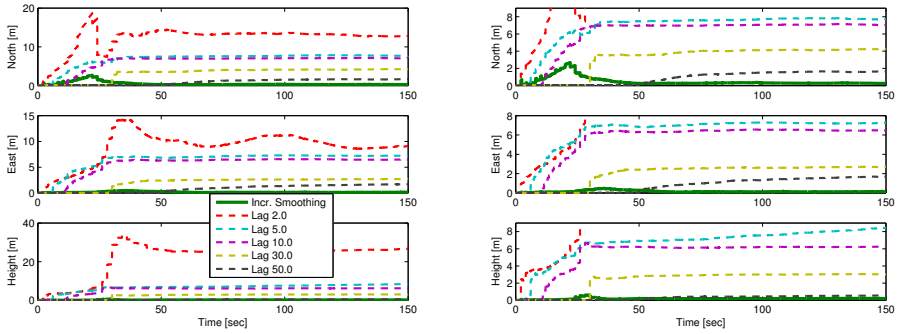


(e) Accelerometer bias



(f) Gyroscope bias

Figure 6: RMSE errors in 100 Monte-Carlo runs: comparison between batch optimization, incremental smoothing, fixed-lag smoother with $2, 5, 10, 30$ and 50 second lags. Incremental smoothing produces nearly identical results as the MAP estimate obtained by a batch optimization, while maintaining small processing time (cf. Figure 5b). Increasing lag size improves performance of fixed-lag smoothers, however has major impact on computational complexity. Performance of fixed-lag smoothers approaches batch optimization using a 50 seconds lag (cf. Figures 6b and 7).



(a) Position

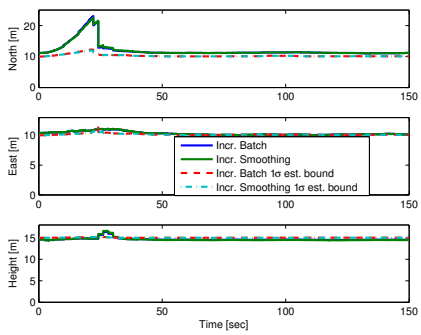
(b) Position - zoom

Figure 7: Root mean square difference with respect to a batch optimization in 100 Monte-Carlo runs: comparison between incremental smoothing and fixed-lag smoother with $2, 5, 10, 30$ and 50 second lags. While the smoothing lag is not full, fixed lag smoothers produce identical results to the MAP estimate (batch optimization), which start to differ afterwards. On a statistical level, the larger the smoothing lag, the smaller is the difference with a batch optimization.

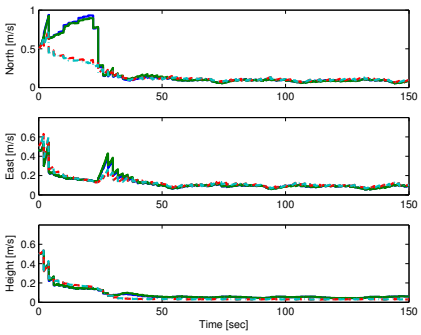
standard stereo visual odometry system was used to detect and match features both between the stereo image pairs and across pairs of images through time. The generated landmark tracks were typically between 2 and 5 frames long, with the longest track lasting 41 frames. The landmark observations were converted into stereo projection factors (25) and added to the factor graph at the frame rate of 10Hz. The KITTI data provided stereo calibration information, so no additional calibration nodes were required for the stereo factors. Only sequential feature tracks were used in this test; no long-term loop closure constraints were incorporated. Figure 9 shows several typical camera images with the tracked features indicated in red.

Figure 10 shows the computed trajectories of full batch optimization, the proposed incremental smoothing method, and fixed-lag smoothers with lag sizes of 0.1, 0.5, 1.0, 2.0, and 10.0 seconds. As can be seen, the proposed incremental smoothing method (green) tracks the MAP estimate (blue) closely over the entire trajectory. The fixed-lag smoothers, on the other hand, show a considerable amount of drift. This is shown in more detail in Figure 11, which shows the root mean squared difference of the position, velocity and orientation of the incremental smoother and various fixed-lag smoothers with respect to the full batch optimization. As can be seen from this figure, the incremental smoothing approach produces results very close to the batch estimate, with the most dramatic differences in the position estimates. In terms of position errors, the incremental smoothing approach produces a maximum error of 2.5m, versus 17m to 38m for the various fixed-lag smoothers.

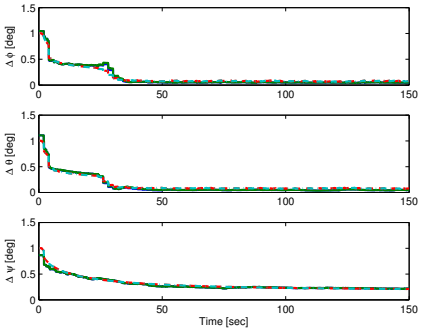
In terms of timing performance, the proposed incremental smoother requires



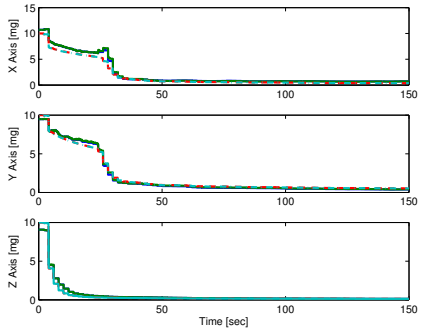
(a) Position



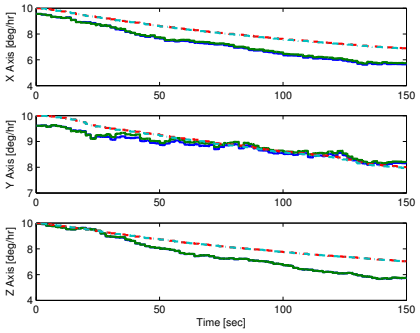
(b) Velocity



(c) Orientation



(d) Accelerometer bias



(e) Gyroscope bias

Figure 8: Further comparison between incremental smoothing and batch optimization. RMSEs and covariance estimates are nearly identical.

Figure 9: Typical camera images from the test KITTI dataset. Features tracked by the visual odometry system are indicated in red.

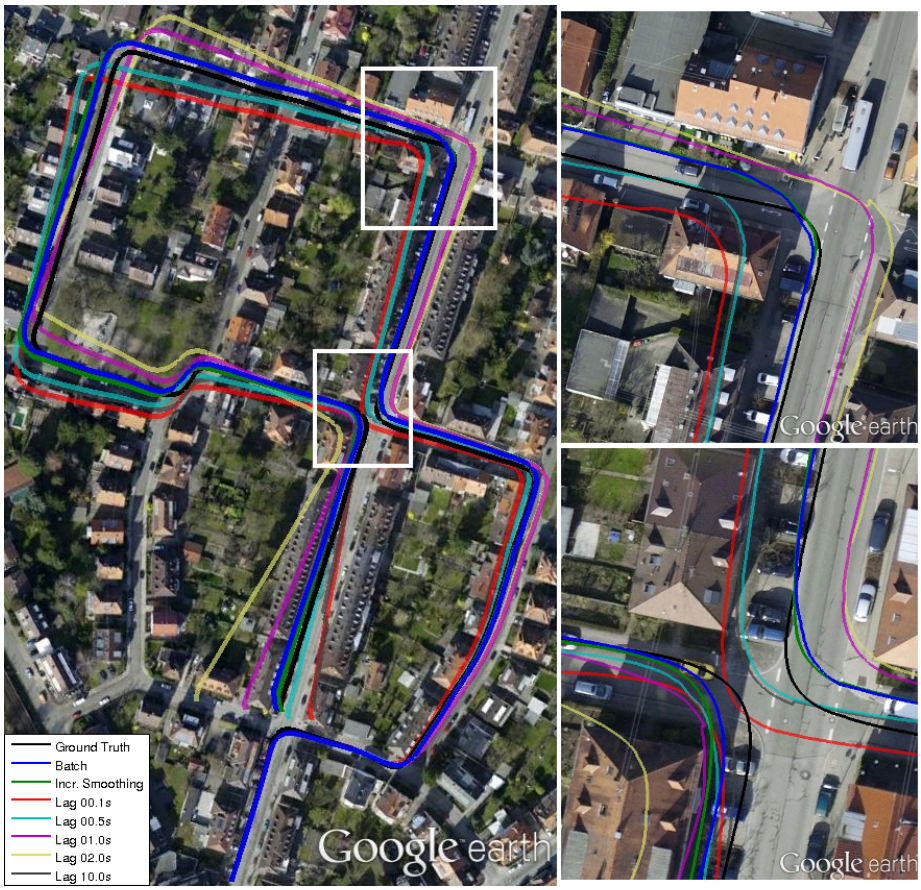
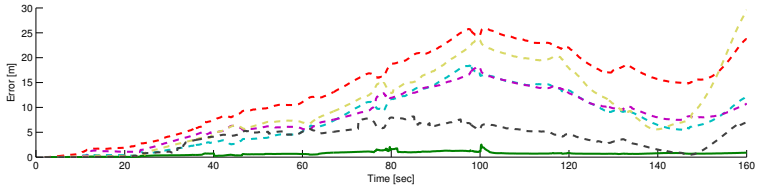
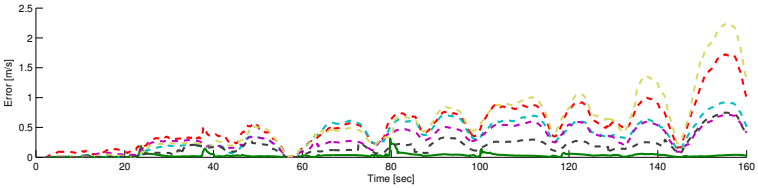


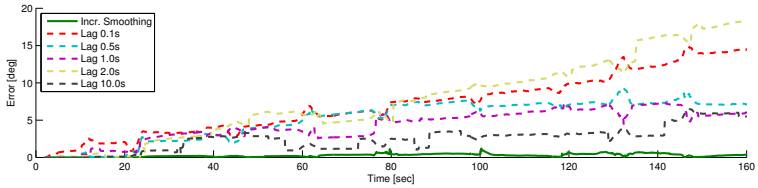
Figure 10: Trajectories for the ground truth, full batch optimization, incremental smoothing, and fixed-lag smoothers with lag sizes of 0.1, 0.5, 1.0, 2.0 and 10.0 seconds.



(a) Position

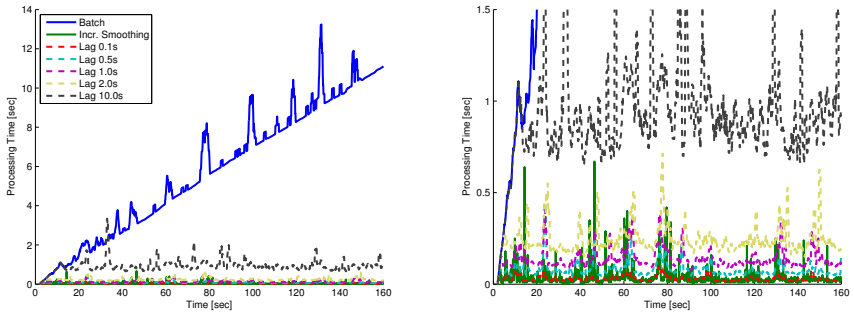


(b) Velocity



(c) Orientation

Figure 11: Root mean square difference of position, velocity and orientation with respect to full batch optimization: comparison between incremental smoothing and fixed-lag smoothers with 0.1, 0.5, 1.0, 2.0 and 10.0 second lags.



(a) Overview (b) Detail

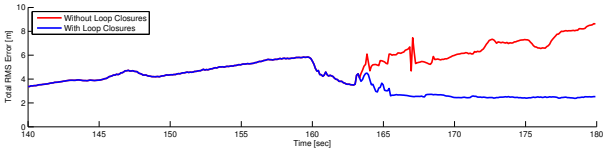
Figure 12: (a) Computational cost of processing the KITTI dataset. (b) Zoom in. For clarity, the presented processing time of all methods, except of incremental smoothing, has been smoothed.

less computation than any of the tested fixed-lag smoothers. Figure 12 shows the required processing time of the incremental smoother and the fixed-lag smoothers. During each update of the fixed-lag smoothers, multiple nonlinear optimization steps may be required before convergence, leading to large fluctuations in the frame-to-frame processing time. To compensate for this affect, the timing results shown in Figure 12 have been averaged over 10 consecutive frames so that the general timing trend is revealed. As shown, the required processing time of the incremental smoother is similar to or less the processing time of the shortest lag smoother, 0.1s lag version, while producing estimates superior to the 10.0s lag smoother.

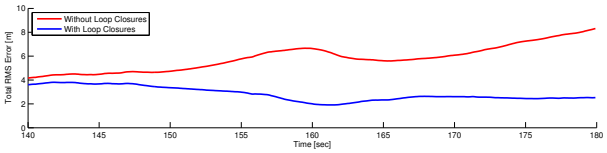
7.3. Loop Closures

In the previous sections, we demonstrated the ability of incremental smoothing to produce a high quality navigation estimate at a high update rate. One further aspect of the incremental smoothing method is that an estimate for the whole trajectory is available at every time step. This allows additional capabilities, perhaps the most compelling of which is the ability to incorporate loop closure constraints at arbitrary locations in the past trajectory. Fixed-lag smoothers can only handle loop closures that occur within the smoother lag. If loop closure handling is desired, this pushes the fixed-lag design to use larger and larger lags, increasing the computational requirements of every update. In contrast, the incremental smoothing approach only consumes additional computation resources during updates that contain loop closure constraints.

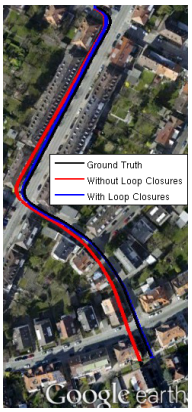
The ability of incremental smoothing to close loops and fully smooth the entire trajectory are briefly highlighted using the same ground-based dataset from the KITTI Vision Benchmark Suite [9]. In the previous section, results were shown using the first 160s of this dataset, which forms a single, large loop.



(a) Intermediate causal position errors



(b) Final smoothed position errors



(c) Final smoothed trajectory

Figure 13: A comparison of the trajectories generated for the KITTI dataset with and without loop closure constraints: (a) position errors of the intermediate causal solution relative to the ground truth, (b) position errors of the final smoothed solution relative to the ground truth, and (c) the final smoothed trajectory.

Between 160s and 170s, the vehicle re-traverses a section of roadway. Approximately 50 landmarks were identified as re-observations of previous landmarks during this re-traverse, and these loop closures were incorporated into the incremental smoother accordingly. Figure 13 shows the position error from the ground truth trajectory with and without the addition of these loop closures. Figure 13a shows the intermediate causal results from the incremental smoother. It can be clearly seen that the first loop closure is added shortly after 160s; before this time the two incremental smoothers are operating with identical measurements. Figure 13b shows the fully smoothed results from the final time at 180s. The incremental smoother adjusted the past trajectory in response to the loop closures, resulting in significantly lower position errors.

However, because a loop closure can involve an arbitrarily large number of past states, the computational requirements of incorporating the loop closure will be unknown at design time. Thus, approaches that process all measurements in sequence, such as incremental smoothing, may not be appropriate when real-time operation is paramount and large loop closures are possible. Several parallel processing data fusion techniques have been introduced recently [18, 25, 26], including a factor-graph based architecture [17] that utilizes incremental smoothing internally.

8. Conclusions

This paper presented a new approach for high-rate information fusion in modern navigation systems based on the available sensors, which typically operate at different frequencies and can be asynchronous. The information fusion problem was formulated using a graphical model, a factor graph, that represents a factorization of the corresponding joint probability distribution function given all the measurements thus far. Such a representation provides a plug and play capability, since measurements from new sensors are easily incorporated into the factor graph using appropriate factors, while sensors that become unavailable are trivially handled. While calculating the maximum a posteriori (MAP) estimate involves computationally expensive batch optimization, a recently-developed incremental inference algorithm, incremental smoothing, was used to produce near-optimal estimates at a fraction of the computational complexity. This algorithm exploits the system sparsity and graph topology to identify the variables that should be optimized when a new measurement becomes available. Thus, only a small portion of variables is recalculated, leading to high-rate performance, as opposed to re-optimizing all the variables in the state vector during every update, as is performed in batch optimization, fixed-lag smoothing and popular variations of the extended Kalman filter. The calculated estimates approach the MAP solution since re-linearization of appropriate variables is possible for any measurement model at any time as no marginalization of past variables occurs. To maintain high-rate performance over time, consecutive IMU measurements were represented by a single non-linear factor, an equivalent IMU factor, that is based on a recently-developed technique for IMU pre-integration, and allows a significant reduction in the number of variables in the optimization.

The proposed method was studied both in a simulated environment and in an experiment. Statistical results, based on a synthetic aerial scenario that involved IMU, magnetometer and monocular camera sensors, showed that the performance of the developed method is very close to a batch optimization, with a much higher computational cost required by fixed-lag smoothers to obtain similar levels of accuracy. The method was validated in an experiment using real IMU and imagery data that was recorded by an autonomous ground vehicle. The obtained performance approached batch optimization while preserving small computational cost.

In general, loop closure measurements are not intended to be processed using the method presented in this paper, since high-rate performance cannot be guaranteed. However, a typical loop-closure scenario is considered to demonstrate the flexibility of the proposed incremental smoothing methodology. Ongoing research focuses on incorporating loop closure information in a parallel process to maintain high-rate updates [17].

Acknowledgements

This material is based upon work supported by the DARPA All Source Positioning and Navigation (ASPN) Program under USAF/ AFMC AFRL Contract FA8650-11-C-7137. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the US government/Department of Defense. The authors would like to acknowledge Richard Roberts, College of Computing, Georgia Institute of Technology, for his assistance and expertise with the GTSAM library.

Appendix A: INS Kinematic Equations

This appendix provides the inertial navigation equations leading to Eq. (15). Assuming some general frame a and denoting by b and i the body and inertial frames, the time derivative of the velocity, expressed in frame a, is given by [7]:

$$\dot{v}^a = R_b^a f^b + g^a - 2\Omega_{ia}^a v^a - \left(\Omega_{ia}^a \Omega_{ia}^a + \dot{\Omega}_{ia}^a\right) p^a, \tag{30}$$

where R_b^a is a rotation matrix transforming from body frame to frame a, f^b is the specific force measured by the accelerometers, p^a is the position vector, evolving according to

$$\dot{p}^a = v^a. \tag{31}$$

The matrix Ω_{ia}^a is defined as

$$\Omega_{ia}^a = \left[\omega_{ia}^a\right]_\times, \tag{32}$$

with ω_{ia}^a being the rotational rate of frame a with respect to the inertial frame i, expressed in frame a, and $[.]_x$ is the skew-symmetric operator, defined for any two vectors q_1 and q_2 as $[q_1]_\times q_2 = q_1 \times q_2$. The vector g^a is the position-dependent gravity acceleration.

The time-evolution for the rotation between frame b and a is given by

$$\dot{R}_b^a = R_b^a \Omega_{ab}^b \tag{33}$$

with $\Omega_{ab}^b = \left[\omega^b\right]_\times$ and ω^b denoting the rotation rate measured by the gyroscopes.

Eqs. (30)-(33) can always be written in the form of Eq. (15), although different expressions are obtained for each choice of the frame a (such as inertial frame, tangent frame, etc.).

Appendix B: Equivalent IMU Factor Equations

This appendix presents the equations for calculating the pre-integrated delta components from consecutive IMU measurements between some two time instances t_i and t_j, and provides expressions for the predicting function h^{Equiv}.

The original formulation of these equations appears in [22], where it was assumed that the robot operates in small areas and uses a low-cost IMU, and therefore, Earth curvature and Earth rotation are neglected.

Here, we extend the formulation given in [22] by taking these two neglected aspects into account. Since the pre-integrated delta components may represent non-negligible rotational motion, we use the underlying Lie algebra (cf. e.g., [1]), with the operators *Expmap* and *Logmap* representing the exponential and logarithm maps over $SE(3)$.

Pre-Integration of IMU Measurements

We assume the starting pre-integration time is t_i and use the notations $\Delta p^{b_i}_{i\to j}$, $\Delta v^{b_i}_{i\to j}$,$R^{b_i}_{b_j}$ to represent the position, velocity and orientation components, respectively, calculated by pre-integrating the IMU measurements from time t_i to some time t_j. In order to avoid recalculating these components when re-linearizing (cf. Section 4.3), Lupton and Sukkarieh [22] perform the integration in the *body* frame of the starting pre-integration time t_i, rather than in the global frame. The body frame at t_i is denoted by b_i. Finally, R^b_a represents a rotation from system a to system b.

The pre-integrated delta components are initialized as: $\Delta p^{b_i}_{i\to i} = 0$, $\Delta v^{b_i}_{i\to i} = 0$ and $R^{b_i}_{bi} = I$. Let $\Delta x_{i\to j} \doteq \left\{\Delta p^{b_i}_{i\to j}, \Delta v^{b_i}_{i\to j}, R^{b_i}_{b_j}\right\}$.

Given the previous pre-integrated components $\Delta p^{b_i}_{i\to j}, \Delta v^{b_i}_{i\to j}, R^{b_i}_{b_j}$, with $t_j \geq t_i$, and the calibration parameters at the starting time t_i (denoted by c_i), the equations for adding a new IMU measurement at time $t_{j+1} \doteq t_j + \Delta t$, comprising the specific force f_j and the angular velocity ω_j, are given in Algorithm 3.

Algorithm 3 Pre-Integration of an IMU measurement f_j, ω_j

1: **Input**:
   IMU measurement f_j, ω_j
   Calibration parameters c_i
   Previous pre-integrated delta components $\Delta x_{i\to j}$
2: Correct IMU measurements with the calibration parameters c_i
3: Position: $\Delta p^{b_i}_{i\to j+1} = \Delta p^{b_i}_{i\to j} + \Delta v^{b_i}_{i\to j}\Delta t$
4: Velocity: $\Delta v^{b_i}_{i\to j+1} = \Delta v^{b_i}_{i\to j} + R^{b_i}_{b_j} f_j \Delta t$
5: Orientation: $R^{b_i}_{b_{j+1}} = R^{b_i}_{b_j} Expmap(\omega_j)$
6: **Output**:
   Pre-integrated delta components $\Delta x_{i\to j+1}$:

$$\Delta x_{i\to j+1} \doteq \left\{\Delta p^{b_i}_{i\to j+1}, \Delta v^{b_i}_{i\to j+1}, R^{b_i}_{b_{j+1}}\right\}$$

Prediction Function h^{Equiv}

The function h^{Equiv} is used to predict the navigation state x_j, given: a) the pre-integrated delta components $\Delta x_{i\to j}$ between the time instances t_i and

t_j, and b) the navigation state x_i and the calibration parameters c_i. Thus, $h^{Equiv} = h^{Equiv}(x_i, c_i, \Delta x_{i\to j})$.

We use the notations $p_i^{L_i}$, $v_i^{L_i}$ and $R_{b_i}^{L_i}$ to represent position, velocity and orientation components in the navigation state x_i. L_i denotes the LLLN[5] frame at time t_i, with the origin defined at the initial position of the robot. The rotation matrix $R_{b_i}^{L_i}$ represents a rotation from the body to LLLN frame at t_i. Recall that the pre-integrated delta components $\Delta x_{i\to j} \doteq \left\{\Delta p_{i\to j}^{b_i}, \Delta v_{i\to j}^{b_i}, R_{b_j}^{b_i}\right\}$ are expressed in the body frame at t_i.

The navigation state x_j, comprising the position, velocity and orientation terms $p_j^{L_j}, v_j^{L_j}$ and $R_{b_j}^{L_j}$ can be calculated as follows.

$$p_j^{L_j} = R_{L_i}^{L_j}\left\{p_i^{L_i} + R_{b_i}^{L_i}\Delta p_{i\to j}^{b_i} + v_i^{L_i}(t_j - t_i) + \left[\frac{1}{2}g^{L_i} - 2\left[\omega_{iL_i}^{L_i}\right]_\times v_i^{L_i}\right](t_j - t_i)^2\right\} \tag{34}$$

$$v_j^{L_j} = R_{L_i}^{L_j}\left\{v_i^{L_i} + R_{b_i}^{L_i}\Delta v_{i\to j}^{b_i} + \left[g^{L_i} - 2\left[\omega_{iL_i}^{L_i}\right]_\times v_i^{L_i}\right](t_j - t_i)\right\} \tag{35}$$

$$R_{b_j}^{L_j} = R_{L_i}^{L_j}\left\{R_{b_i}^{L_i} Expmap\left(\Delta\varphi\right)\right\} \tag{36}$$

where

$$\Delta\varphi = Logmap\left(R_{b_j}^{b_i}\right) - R_{L_i}^{b_i}\omega_{iL_i}^{L_i}(t_j - t_i), \tag{37}$$

and g^{L_i} is the gravity vector, which is assumed to be constant between t_i and t_j. The vector $\omega_{iL_i}^{L_i}$ represents the rotational rate of the LLLN frame with respect to the inertial frame i (cf. Appendix A). Finally, it is often reasonable to assume the distance travelled between t_i and t_j is not too large so that $R_{L_i}^{L_j} \approx I$.

[1] M. Agrawal. A Lie algebraic approach for consistent pose registration for motion estimation. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[2] Mitch Bryson, M. Johnson-Roberson, and Salah Sukkarieh. Airborne smoothing and mapping using vision and inertial sensors. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 3143–3148, 2009.

[3] T.A. Davis, J.R. Gilbert, S.I. Larimore, and E.G. Ng. A column approximate minimum degree ordering algorithm. *ACM Trans. Math. Softw.*, 30(3):353–376, 2004. ISSN 0098-3500.

[4] F. Dellaert and M. Kaess. Square Root SAM: Simultaneous localization and mapping via square root information smoothing. *Intl. J. of Robotics Research*, 25 (12):1181–1203, Dec 2006.

[5] LLLN denotes the local-level local-north coordinate system [7]. For long-term navigation, it is better to represent the position term in a more appropriate coordinate system, such as ECEF or LLH, as commonly done in the inertial navigation literature.

[5] D. Diel, P. DeBitetto, and S. Teller. Epipolar constraints for vision-aided inertial navigation. In *Proceedings of the IEEE Workshop on Motion and Video Computing (WACV/MOTION'05)*, Washington, DC, USA, 2005.

[6] R.M. Eustice, H. Singh, and J.J. Leonard. Exactly sparse delayed-state filters for view-based SLAM. *IEEE Trans. Robotics*, 22(6):1100–1114, Dec 2006.

[7] J.A. Farrell. *Aided Navigation: GPS with High Rate Sensors*. McGraw-Hill, 2008.

[8] J. Folkesson, P. Jensfelt, and H.I. Christensen. Graphical SLAM using vision and the measurement subspace. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Aug 2005.

[9] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Providence, USA, June 2012.

[10] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[11] P. Heggernes and P. Matstoms. Finding good column orderings for sparse QR factorization. In *Second SIAM Conference on Sparse Matrices*, 1996.

[12] G.P. Huang, A.I. Mourikis, and S.I. Roumeliotis. An observability-constrained sliding window filter for SLAM. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 65–72, 2011.

[13] V. Indelman, P. Gurfil, E. Rivlin, and H. Rotstein. Real-time vision-aided localization and navigation based on three-view geometry. *IEEE Trans. Aerosp. Electron. Syst.*, 48(3):2239–2259, July 2012.

[14] V. Indelman, S. Wiliams, M. Kaess, and F. Dellaert. Factor graph based incremental smoothing in inertial navigation systems. In *Intl. Conf. on Information Fusion, FUSION*, 2012.

[15] M. Kaess, V. Ila, R. Roberts, and F. Dellaert. The Bayes tree: An algorithmic foundation for probabilistic robot mapping. In *Intl. Workshop on the Algorithmic Foundations of Robotics*, Dec 2010.

[16] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *Intl. J. of Robotics Research*, 31:217–236, Feb 2012.

[17] M. Kaess, S. Wiliams, V. Indelman, R. Roberts, J.J. Leonard, and F. Dellaert. Concurrent filtering and smoothing. In *Intl. Conf. on Information Fusion, FUSION*, 2012.

[18] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *IEEE and ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR)*, pages 225–234, Nara, Japan, Nov 2007.

[19] X. Kong, E. M. Nebot, and H. Durrant-Whyte. Development of a nonlinear psi-angle model for large misalignment errors and its application in INS alignment and calibration. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1430–1435, 1999.

[20] F.R. Kschischang, B.J. Frey, and H-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. Inform. Theory*, 47(2), February 2001.

[21] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, pages 333–349, Apr 1997.

[22] T. Lupton and S. Sukkarieh. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Trans. Robotics*, 28(1):61–76, Feb 2012.

[23] Y. Ma, S. Soatto, J. Kosecka, and S.S. Sastry. *An Invitation to 3-D Vision*. Springer, 2004.

[24] A.I. Mourikis and S.I. Roumeliotis. A multi-state constraint Kalman filter for vision-aided inertial navigation. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 3565–3572, April 2007.

[25] A.I. Mourikis and S.I. Roumeliotis. A dual-layer estimator architecture for long-term localization. In *Proc. of the Workshop on Visual Localization for Mobile Platforms at CVPR*, Anchorage, Alaska, June 2008.

[26] R.A. Newcombe, S.J. Lovegrove, and A.J. Davison. DTAM: Dense tracking and mapping in real-time. In *Intl. Conf. on Computer Vision (ICCV)*, pages 2320–2327, Barcelona, Spain, Nov 2011.

[27] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[28] G. Sibley, L. Matthies, and G. Sukhatme. Sliding window filter with application to planetary landing. *J. of Field Robotics*, 27(5):587–608, 2010.

[29] J.P. Tardif, M. George, M. Laverne, A. Kelly, and A. Stentz. A new approach to vision-aided inertial navigation. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4161–4168, 2010.

[30] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT press, Cambridge, MA, 2005.

[31] N. Trawny, A. I. Mourikis, S. I. Roumeliotis, A. E. Johnson, and J. F. Montgomery. Vision-aided inertial navigation for pin-point landing using observations of mapped landmarks: Research articles. *J. of Field Robotics*, 24(5):357–378, May 2007.