Incremental Light Bundle Adjustment for Robotics Navigation

Vadim Indelman, Andrew Melim, and Frank Dellaert

Abstract— **This paper presents a new computationally-efficient method for vision-aided navigation (VAN) in autonomous robotic applications. While many VAN approaches are capable of processing incoming visual observations, incorporating loop-closure measurements typically requires performing a bundle adjustment (BA) optimization, that involves both all the past navigation states and the observed 3D points. Our approach extends the incremental light bundle adjustment (LBA) method, recently developed for structure from motion [10], to information fusion in robotics navigation and in particular for including loop-closure information. Since in many robotic applications the prime focus is on navigation rather then mapping, and as opposed to traditional BA, we algebraically eliminate the observed 3D points and do not explicitly estimate them. Computational complexity is further improved by applying incremental inference. To maintain high-rate performance over time, consecutive IMU measurements are summarized using a recently-developed technique and navigation states are added to the optimization only at camera rate. If required, the observed 3D points can be reconstructed at any time based on the optimized robot's poses. The proposed method is compared to BA both in terms of accuracy and computational complexity in a statistical simulation study.**

I. Introduction

The increasing usage of autonomous robots to carry out different types of tasks has urged development of advanced methods for reliable long-term localization and navigation. In particular, navigation in unknown environments has been at the focus of many research efforts for the past two decades. Indeed, many simultaneous localization and mapping (SLAM) and navigation-aiding methods have been developed over the years.

One of the key challenges in long-term operation is ever-growing computational complexity. Typically, in order to incorporate loop-closure measurements (such as in the scenario shown in Figure 1), new variables are constantly added to the estimation process, which can be considered as a nonlinear optimization. These usually include the robot's navigation state, as well as the observed 3D points. The optimization, also known as bundle adjustment (BA) in computer vision and full SLAM in robotics, is thus performed over a large set of variables, and while efficient incremental optimization approaches have been recently developed, reducing the computational complexity is important for many practical robotic applications.

In this paper we address this challenge and propose a new method for navigation aiding in which the number of variables is significantly reduced. Motivated by the observation that in many robotic applications the prime focus is on navigation rather on the mapping, we suggest a method that reduces the variable count in the optimization by avoiding explicit estimation of the observed 3D points.

The authors are with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA. emails: indelman@cc.gatech.edu,{andrew.melim, dellaert}@gatech.edu.

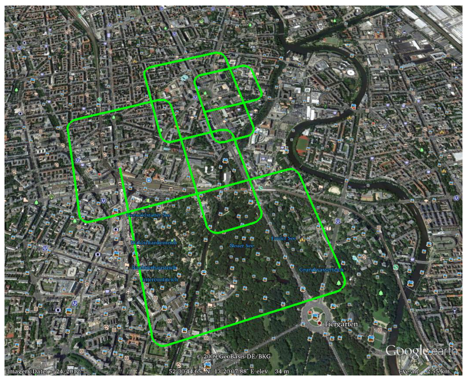


Fig. 1: Trajectory used for analyzing processing time, shown for illustration purposes in Google Earth.

In our previous work [10], we showed in the context of structure from motion (SfM), that algebraic elimination of 3D points allows substantial improvement in the computational complexity while maintaining similar levels of accuracy. While the proposed method avoids explicitly estimating the observed map, this map or any part of it can be always reconstructed, if required, based on the optimized robot's navigation states [10]. Here, we extend that work to robotics navigation where additional sensors are usually available and images are provided in a known order. Specifically, we consider the challenging setup of an inertial measurement unit (IMU) and a monocular camera. We also present the use of pre-integration methods developed in [21] to reduce computational complexity when used to initialize camera positions in a navigation context.

This paper presents the following contributions: a) Reduced computational complexity and similar levels of accuracy performance as full bundle adjustment ; b) Loop-closure capable vision-aided navigation; c) Method validation using a statistical simulation study.

The remainder of this paper is organized as follows. After discussing related work, we formulate the robotics navigation problem and present our general approach in Section III. Section IV reviews the incremental light bundle adjustment method, which is then extended to robotics navigation in Section V. Statistical simulation results comparing the method to full bundle adjustment, as well as a comparison of processing time, are given in Section VI. Main conclusions and future work are suggested in Section VII.

II. Related Work

Early SLAM methods [24], [25] applied the extended Kalman filter (EKF) to simultaneously estimate the current robot's pose and the observed 3D points. However due to its quadratic computational complexity and since it was realized that marginalizing out past poses in EKF-SLAM causes fill-in in the information matrix [26], [2], recent SLAM approaches use smoothing for inference. Today SLAM algorithms can be divided into two main categories: feature- and view-based SLAM.

In feature-based SLAM, both the observed 3D points and the robot's past and current poses are optimized. Several efficient optimization methods that exploit the sparsity of typical SfM and SLAM problems have been developed in recent years, including [19], [16] and [14]. Lourakis et al. [19] perform efficient optimization by exploiting the sparse connections between the pose and the landmark variables, while in [16] the inter-pose sparsity is also taken into account. As opposed to the commonly applied batch optimization, the incremental smoothing and mapping approach [14], also used in this paper, recovers the exact solution while recalculating only part of the variables each time a new measurement is added into the optimization, resulting in a significant reduction of computational cost.

The second SLAM category is view-based SLAM [20], [2], or pose-SLAM, in which, similar to the proposed method, only the current and past robot's poses are maintained. In pose-SLAM approaches, pairs of poses are linked using relative pose constraints that are straightforward to estimate in a stereo camera setup [7], [17], but become more challenging when relying only on a single camera. In the latter case, the relative constraints can be estimated only up to a scale, which encodes the magnitude of the relative translation [2]. This scale parameter can be set based on the previous frames as in [1]. However, to avoid scale drift the scale parameters should be part of the optimization as well [4]. In contrast to conventional pose-SLAM, we formulate multi-view geometry constraints for *each* feature match, thereby not relying on uncertainty estimates obtained by the intermediate (and separate) process of image-based relative-pose constraints estimation.

In the context of navigation-aiding, despite the close relation to SLAM, only a few methods have been presented in recent years that are capable of incorporating loop closure measurements. These include [22] where visual observations are incorporated into the navigation solution using an EKF formulation with a sliding window of past poses. In a later work [23], the authors applied a conventional batch BA that involved explicit structure estimation in order to handle loop closure measurements.

More recently, incremental smoothing [14] was proposed for inertial navigation systems in [12], [13] and a method was developed to incorporate loop closures while maintaining a real time navigation solution [15]. Our method is formulated within the same framework of [15], [12], [13] but replaces the explicit estimation of the 3D points with a set of 2-view and 3-view constraints. Other work involving the use of incremental smoothing in a navigation context was presented in [6] in which Doppler Velocity Log (DVL), and IMU measurements were integrated with sonar and monocular camera measurements to perform real-time SLAM relative to the hull of large ships.

III. Problem Formulation and Approach

In this paper we consider a robotics navigation application, where measurements from different sensors should be fused, and focus on IMU and visual monocular sensors, a commonly used configuration. Denote the navigation state, comprising the robot's pose and velocity, at time t_k by x_k and the set of past and current navigation states, up to time t_k by $X_k \doteq \begin{bmatrix} x_1^T & \dots & x_k^T \end{bmatrix}^T$. Also, let Z_k represent all the available measurements up to time t_k, which includes both the IMU measurements and the visual observations of 3D points. We denote the set of observed 3D points until time t_k by L_k.

SLAM and the corresponding robot navigation problem can be described referring to the joint probability distribution function $p\left(X_k, L_k|Z_k\right)$. Full SLAM calculates the maximum a posteriori (MAP) estimate of both the robot past and current state and the map

$$X_k^*, L_k^* = \arg\max_{X_k, L_k} p\left(X_k, L_k|Z_k\right),$$

which corresponds, conventionally assuming Gaussian distributions, to a non-linear least square optimization. As seen, both the robot's navigation states X_k and the observed 3D points L_k are involved, and thus if $X_k \in \mathbb{R}^{N_k \times 1}$ and $L_k \in \mathbb{R}^{M_k \times 1}$, then there are $N_k + M_k$ variables in the optimization at time t_k.

However, in the context of navigation, we only want the density of X_k, and mainly the density of the current navigation state x_k. This could be done by marginalizing out the 3D points

$$p\left(X_k|Z_k\right) = \int p\left(X_k, L_k|Z_k\right) dL_k, \qquad (1)$$

however that requires the joint $p\left(X_k, L_k|Z_k\right)$ to be available and is expensive to calculate.

Instead, inspired by a recently-developed approach, incremental light bundle adjustment (iLBA) [10], we propose to approximate $p\left(X_k|Z_k\right)$ by algebraically eliminating the observed 3D points. The resulting probability distribution function (pdf), that we denote by $p_{LBA}\left(X_k|Z_k\right)$ and further discuss in the next section, does not involve 3D points, and thus the corresponding optimization is performed over only N_k variables. Applying incremental smoothing [14] only part of these variables are actually recalculated each time a new measurement is added.

Optimizing only over the navigation states (and the IMU calibration parameters, as discussed in Section V) results in reduced computational complexity, while the actual accuracy, although somewhat degraded with respect to BA, is sufficient in many cases. Avoiding having the landmarks in the optimization eliminates the need for landmark initialization, which is often not trivial, as well as having less tuning parameters in the incremental smoothing approach.

Furthermore, since IMU measurements are high-rate, the number of navigation states N_k rapidly increases with time and high-rate performance quickly becomes infeasible even when optimizing only $p_{LBA}\left(X_k|Z_k\right)$. To address this challenge, we adopt a recently-developed technique for IMU pre-integration [21], and incorporate navigation states into the optimization only at camera rate.

IV. Incremental Light Bundle Adjustment

Incremental light bundle adjustment (iLBA) [10] combines the following two key-ideas: algebraic elimination of 3D points, and incremental smoothing. In this section we review each of these concepts.

A. Algebraic Elimination of 3D points

The joint pdf $p\left(X_k, L_k|Z_k\right)$ can be explicitly written in terms of the prior information and the actual process and measurement models. If we consider for a moment only the visual observations, then:

$$p\left(X_k, L_k|Z_k\right) = priors \cdot \prod_i \prod_j p\left(z_i^j | x_i, l_j\right),$$

where we denote the measured image observation of the j^{th} 3D point l_j in the ith image by z_i^j. Assuming Gaussian distributions, MAP estimation corresponds to the following nonlinear optimization

$$J_{BA}\left(X_k, L_k\right) = \sum_i \sum_j \left\| z_i^j - proj\left(x_i, l_j\right)\right\|_\Sigma^2, \quad (2)$$

where $proj\left(.\right)$ is the projection operator [5] for a standard pinhole camera model, and $\|a\|_\Sigma^2 \doteq a^T \Sigma^{-1} a$ is the squared Mahalanobis distance with the measurement covariance matrix Σ.

Considering the robot's poses that observe some common 3D point l and writing down all the appropriate projection equations, it is possible to algebraically eliminate l which results in constraints between triplets of poses [27], [10]. One possible formulation of these constraints, recently developed in the context of vision-aided navigation [8], [9], are the three-view constraints. Assuming three overlapping views k, l and m, these constraints are

$$g_{2v}\left(x_k, x_l, z_k, z_l\right) = q_k \cdot \left(t_{k\to l} \times q_l\right) \quad (3)$$

$$g_{2v}\left(x_l, x_m, z_l, z_m\right) = q_l \cdot \left(t_{l\to m} \times q_m\right) \quad (4)$$

$$g_{3v}\left(x_k, x_l, x_m, z_k, z_l, z_m\right) = \quad (5)$$
$$\left(q_l \times q_k\right) \cdot \left(q_m \times t_{l\to m}\right) - \left(q_k \times t_{k\to l}\right) \cdot \left(q_m \times q_l\right)$$

where $q_i \doteq R_i^T K_i^{-1} z$ for any view i and image observation z, K_i is the calibration matrix of this view, R_i represents the rotation matrix from some arbitrary global frame to the i^{th} view's frame, and $t_{i\to j}$ denotes the translation vector from view i to view j, expressed in the global frame. The first two constraints are the two-view constraints g_{2v} between appropriate pairs of views, while the third constraint, g_{3v}, involves all the three views.

When a 3D point is observed by more than three views, we add a single two-view and three-view constraint between each new view and past views, as further explained in [10].

Consequently, rather than optimizing the bundle adjustment cost function (2), that involves both the pose and landmark variables, in light bundle adjustment (LBA) the cost function is [10]:

$$J_{LBA}(X) \doteq \sum_{i=1}^{N_h} \|h_i(X_i, Z_i)\|_{\Sigma_i}^2, \quad (6)$$

where h_i represents a single two- or three-view constraint ($h_i \in \{g_{2v}, g_{3v}\}$) that is a function of several views $X_i \subset X$ and image observations Z_i in these views, and N_h is the overall number of such constraints.

The LBA cost function (6) corresponds to a probability distribution $p_{LBA}\left(X|Z\right)$, which approximates the pdf $p\left(X|Z\right)$ from Eq. (1) [11]. One can observe that Eq. (6) indeed does not contain any structure parameters, and hence the overall number of variables in the optimization is significantly reduced compared to the bundle adjustment cost function (2).

B. Incremental Smoothing

The second component in iLBA is incremental smoothing [14], which re-calculates only part of the robot's poses each time a new measurement is incorporated into the optimization. Although a detailed description of the incremental smoothing approach is beyond the scope of this paper, we briefly discuss the essentials next.

The factorization of the joint pdf into individual process and measurement models can be represented using a graphical model, known as the factor graph [18], upon which incremental smoothing is performed.

In the context of LBA, the probability distribution $p_{LBA}\left(X|Z\right)$ can be factorized as

$$p_{LBA}\left(X|Z\right) \propto \prod_{i=1}^{N_h} f_{2v/3v}\left(X_i\right), \quad (7)$$

with $f_{2v/3v} \in \{f_{2v}, f_{3v}\}$. The two- and three-view factors f_{2v} and f_{3v}, respectively, are defined for the three-view constraints (3)-(5) as

$$f_{2v}\left(x_k, x_l\right) \doteq \exp\left(-\frac{1}{2}\left\|g_{2v}\left(x_k, x_l, z_k, z_l\right)\right\|_{\Sigma_{2v}}^2\right), \quad (8)$$

and

$$f_{3v}\left(x_k, x_l, x_m\right) \doteq \quad (9)$$
$$\exp\left(-\frac{1}{2}\left\|g_{3v}\left(x_k, x_l, x_m, z_k, z_l, z_m\right)\right\|_{\Sigma_{3v}}^2\right),$$

with appropriate covariance matrices Σ_{2v} and Σ_{3v} [10]. In practice, in order to avoid the trivial solution of zero translation, each of the constraints g_{2v} and g_{3v} are normalized by a translation vector and the Jacobian matrices are modified

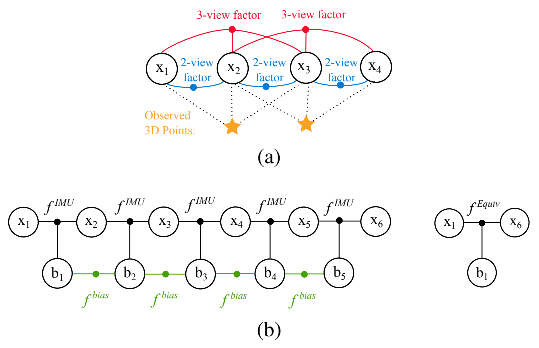


Fig. 2: Factor graph representations: (a) iLBA: incorporating two- and three-view factors instead of projection factors algebraically eliminates the observed 3D points from the optimization; (b) Inertial navigation based on IMU measurements and modeling IMU bias dynamics using conventional and equivalent IMU factors.

accordingly. Figure 2a illustrates a simple factor graph that corresponds to Eq. (7), with 4 camera poses observing 2 different 3D points.

In incremental smoothing, the factor graph is converted into a directed junction tree, which represents a factorization of the square root information matrix $\mathcal{I} = A^T A$ at a given linearization point. Batch optimization performs factorization from scratch each time a new measurement is received. In contrast, incremental smoothing updates the previous factorization with the new information contributed by the received measurements, eliminating only part of the variables. Incremental smoothing performs selective linearization of variables that exceed certain thresholds, typically using different threshold for each variable type (e.g. position, bias etc.) [14]. Choosing appropriate thresholds can be thus considered as a tuning process, since too loose thresholds might result in degraded performance but too tight thresholds will lead to high computational time. In contrast to BA, LBA does not involve thresholds for the landmark variables, resulting in an easier tuning process.

V. Incremental LBA for Robotics Navigation

While iLBA considers only visual observations, in actual robotic applications different additional sensors are usually available. Combining all the available measurements from different sensors then becomes a matter of information fusion and can be handled in the very same framework of incremental smoothing [12].

Rather than focusing on robot localization we consider the more complete problem of navigation, where parameters such as velocity and inertial measurement unit (IMU) calibration parameters are estimated as well. An additional challenge is being able to incorporate the high-rate IMU measurements into the optimization, while maintaining high-rate performance.

Next we discuss how to incorporate IMU and visual measurements using the iLBA concept, and adapt a recently-developed technique for summarizing consecutive IMU measurements to avoid adding navigation variables into the optimization at IMU frequency. Denote the IMU bias by b.

A. *iLBA and Inertial Navigation Process*

The time evolution of the navigation state x can be described as a set of nonlinear differential equations [3]

$$\dot{x} = h_c^{IMU}\left(x, b, z_k^{IMU}\right),$$

where z^{IMU} is the IMU measurement. Different numerical schemes, ranging from a simple Euler integration to high-order Runge-Kutta integration, can be applied for solving these equations. In this work we use a simple Euler integration prediction function with an associated integration uncertainty, allowing the underlying nonlinear optimization in incremental smoothing to adjust the individual state estimates appropriately. The discrete representation is thus [12]:

$$x_{k+1} = h^{IMU}\left(x_k, b_k, z_k^{IMU}\right).$$

In a similar manner, the dynamics of the IMU bias can be described as $b_{k+1} = h^b\left(b_k\right)$ for some known function h^b. For example, h^b can be represent a first order Gauss-Markov process.

The corresponding factors to the IMU process and the IMU bias process are defined as follows:

$$f^{IMU}\left(x_{k+1}, x_k, b_k\right) \doteq \exp\left(-\frac{1}{2}\left\|x_{k+1} - h^{IMU}\left(x_k, b_k, z_k^{IMU}\right)\right\|^2_{\Sigma_{IMU}}\right) \tag{10}$$

$$f^{bias}\left(b_{k+1}, b_k\right) \doteq \exp\left(-\frac{1}{2}\left\|b_{k+1} - h^b\left(b_k\right)\right\|^2_{\Sigma_b}\right)$$

with Σ_{IMU} and Σ_b denoting the corresponding process covariance matrices.

Incorporating IMU measurements and three-view constraints, while taking causality into account, is equivalent to the following overall joint probability distribution function and its factorization

$$p\left(X_k, B_k | Z_k\right) \propto \prod_{s=0}^{k-1}\left[f^{IMU}\left(x_{s+1}, x_s, b_s\right) f^{bias}\left(b_{s+1}, b_s\right) \prod_{i=1}^{n_{s+1}} f_{2v/3v}\left(X_{s_i}\right)\right]$$

with $B_k \doteq \left[\begin{array}{ccc} b_1^T & \cdots & b_k^T \end{array}\right]^T$ being the overall set of IMU calibration parameters. In practice, since these tend to only have slow dynamics, it makes sense to describe this process in some lower rate [12].

In the above equation, n_{s+1} is the number of two- and three-view factors that are added between each current state x_{s+1} and past states. Thus, if $x_a \in X_{s_i}$ then $a \leq s+1$.

The MAP estimate

$$X_k^*, B_k^* = \arg\max_{X_k, B_k} p\left(X_k, B_k | Z_k\right)$$

is obtained by applying incremental smoothing, as mentioned in Section IV-B.

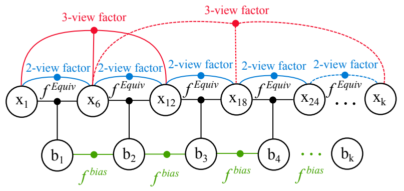


Fig. 3: Factor graph representation for iLBA for robotics navigation - inertial navigation is aided by two- and three-view factors, which are added only between camera poses. Loop closures are incorporated using the same factors (dashed lines). Best viewed in color.

B. Equivalent IMU Factor

To reduce the number of variables and factors in the optimization we adopt a recently-developed technique [21] 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, represented respectively by $\Delta p_{i\to j}, \Delta v_{i\to j}$ and the rotation matrix R_j^i:

$$\Delta x_{i\to j} \doteq \left\{\Delta p_{i\to j}, \Delta v_{i\to j}, R_j^i\right\} = \eta\left(Z_{i\to j}^{IMU}, b_i\right),$$

where $Z_{i\to j}^{IMU}$ is the set of IMU measurements between the time instances t_i and t_j, that are corrected using the bias b_i, and η is a known non-linear function that describes the IMU measurements pre-integration process. One can now use $\Delta x_{i\to j}$ to predict x_j based on the current estimate of x_i. Let h^{Equiv} represent this predicting function.

We can then define an equivalent IMU factor f^{Equiv} as

$$f^{Equiv}\left(x_j, x_i, b_i\right) \doteq \exp\left(-\frac{1}{2}\left\|x_j - h^{Equiv}\left(x_i, b_i, \Delta x_{i\to j}\right)\right\|_\Sigma^2\right), \quad (11)$$

which involves only the variables x_j, x_i and b_i for any reasonable[1] two time instances t_i and t_j. Figure 2b illustrates the conceptual difference between the conventional and equivalent IMU factors.

A straightforward approach for calculating $\Delta x_{i\to j}$ involves pre-integrating the IMU measurements while expressing them in the navigation frame. However, this will require re-calculating $\Delta x_{i\to j}$ from scratch each time the rotation estimate changes, i.e. each re-linearization of x_i. To resolve this, as proposed in [21], the different components in $\Delta x_{i\to j}$ are expressed in the body frame of the first time instant (i.e. t_i), which allows re-linearizing the factor (11) without recalculating $\Delta x_{i\to j}$. The reader is referred to [21] for further details.

[1]The original derivation in [21] neglects Earth curvature and Earth rotation, however it can be extended to the more general case which assumes the gravity vector and the rotation rate of the navigation frame with respect to an inertial frame are constant. The time instances t_i, t_j should be chosen such that these assumptions are satisfied.

The equivalent IMU factor allows to significantly reduce the number of variables and factors in the optimization, and enables high-rate performance while using efficient optimization techniques. This is illustrated in Figure 3, that shows a factor graph with two- and three-view factors and the equivalent IMU factor bridging between navigation states (variables) from different time instances. Note that a conventional IMU factor would require adding consecutive navigation states to the graph.

Since in typical navigation systems a navigation solution x_t is required in real time, i.e. each time an IMU measurement is obtained, one can predict x_t using the accumulated component $\Delta x_{i\to j}$ and the current estimates $\hat{x}_i, \hat{b}_i$ of x_i and b_i, in a parallel process and *without* incorporating x_t into the optimization, i.e. $h^{Equiv}\left(\hat{x}_i, \hat{b}_i, \Delta x_{i\to j}\right)$.

C. Choosing Past Frames for New Multi-View Constraints

An important aspect in LBA is how to choose past camera frames when adding new multi-view factors into the optimization: Assume a new image is captured at time t_k and feature correspondences with past images are established. Consider one such correspondence and denote by $\mathcal{W}$ the time indices in which the same landmark was observed. Adding a two-view factor $f_{2v}\left(x_k, x_l\right)$ requires choosing the past frame l from the set $\mathcal{W}$, and similarly adding a new three-view factor $f_{3v}\left(x_k, x_m, x_n\right)$ involves determining the past frames m and n from the set $\mathcal{W}$. In general, each of the frames m and n can be different than l.

Different approaches for determining these past camera frames yield different graph topologies and are expected to affect both accuracy and computational complexity of the proposed method. While a methodological method for determining these past frames still remains an open question and will be investigated in future research, we have observed that since in the robotics case the captured images are time-ordered, a good heuristic is to set l and n to the earliest frame in $\mathcal{W}$, while the second view in the three-view factor (m) is set such that the magnitudes of relative translations $t_{k\to m}$ and $t_{m\to n}$ are similar.

VI. Results

The proposed method was examined in a simulated realistic aerial scenario, covering an area of about 2×1.5 km, with a footprint of a university campus, as shown in Figure 1. A statistical performance study was conducted using a smaller scenario (Figure 4). In both cases, the aerial vehicle gradually explores different areas and occasionally re-visits previously observed locations thereby providing loop closure measurements. In these scenarios, the flight is at a constant height of 200 meter above mean ground level, with a 40 m/s velocity. A medium-grade IMU and a single downward-facing camera, operating at 100 Hz and 0.5 Hz, were used.

In the following we compare LBA to the BA approach, using the equivalent IMU factors and incremental smoothing in both cases. The two methods were implemented using the GTSAM factor graph optimization library[2]. All the reported

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

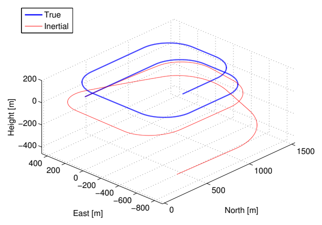


Fig. 4: Trajectory used in Monte-Carlo runs and inertial navigation drift in a typical run.

results were obtained on an Intel i7-2600 processor with a 3.40GHz clock rate and 16GB of RAM, using a single-threaded implementation.

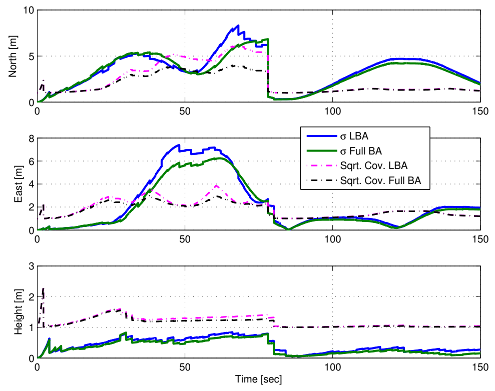
The 100 Hz ideal IMU measurements were corrupted with a constant bias and a zero-mean Gaussian noise in each axis. Bias terms were drawn from a zero-mean Gaussian distribution with a standard deviation of $\sigma = 10$ mg for the accelerometers and $\sigma = 10$ deg/hr for the gyroscopes. The noise terms 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. Visual observations of unknown 3D points were corrupted by a zero-mean Gaussian noise with $\sigma = 0.5$ pixels.

A. *Statistical Simulation Results*

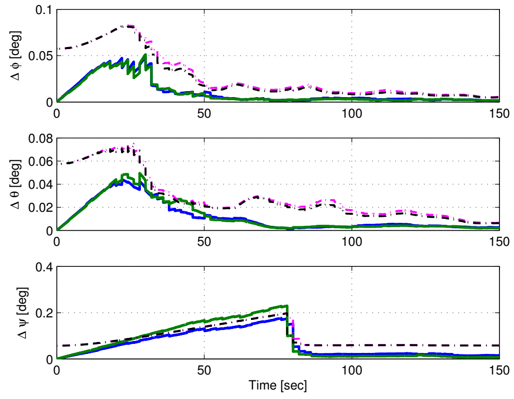
We first compare the performance of the proposed method (iLBA) and full BA based on a 50-run Monte-Carlo study. Actual processing timing for each of the method is analyzed in a larger scenario in Section VI-B. The trajectory used in this study contains several loop closures (at $t \approx 80$ and $t \approx 135$ seconds) during which the same ground areas are re-observed (Figure 4).

The results are given in Figures 5-6 in terms of standard deviation error (1σ) and the uncertainty covariance, while a typical inertial navigation drift is shown in Figure 4. Overall, similar performance is obtained both for iLBA and full BA: Position errors are confined within $5 - 10$ meters in all axes, with a significant reset upon loop closure around $t = 80$ seconds; pitch and roll errors are bounded, while yaw angle error is reduced to prior values upon loop closure; accelerometer bias is estimated during the first segment of straight flight, while gyroscope bias is gradually estimated during maneuvers and loop closure. The covariance in iLBA tends to be a bit higher than the one in full BA.

One should note that the initial navigation errors were assumed to be zero in the presented MC runs. While this is appropriate in many robotics applications that focus on navigation relative to the initial robot position, in other navigation applications, including aerial navigation-aiding, the navigation solution is expressed with respect to some other global frame. Although in the latter case the initial navigation errors might be non-zero, the proposed method is still capable of reducing the development of navigation errors (not shown due to lack of space).



(a)



(b)

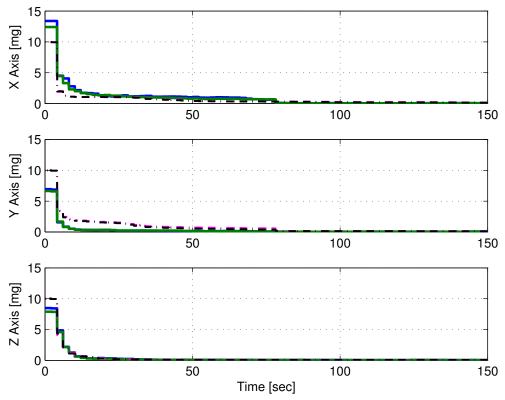
Fig. 5: Monte-Carlo study comparing between LBA and full SLAM, both using incremental smoothing: (a) Position errors; (b) Euler angle errors.

B. *Larger Scenario*

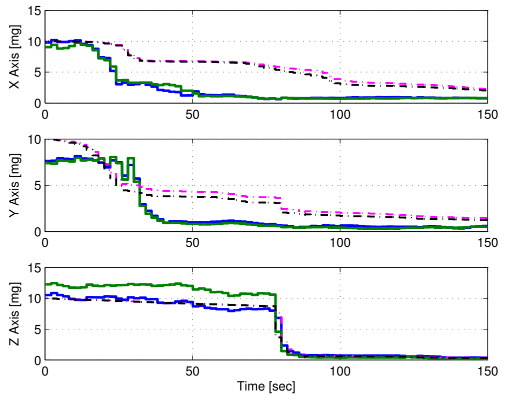
We investigate the computational complexity of LBA and compare it to BA in a larger scenario that is shown in Figure 1. This scenario contains occasional loop closures, some of which are large loop closures, involving many navigation and landmark variables recalculated during optimization. The aerial vehicle travels a total distance of about 13 km in 700 seconds.

A top view of the estimated trajectory by LBA and BA, compared to ground truth and to pure IMU integration, is shown in Figure 7a, with position estimation errors given in Figure 7b. One can observe the fast drift of IMU-based dead reckoning, while both LBA and BA yield estimates close to ground truth with similar levels of accuracy. Note that only IMU and monocular cameras are used, *without* GPS or any additional sensors, producing position estimates with a typical estimation error of $5 - 10$ meters, with a highest estimation error of 20 meters.

While a similar estimation accuracy was obtained both by LBA and BA, processing time is different. The latter depends on the number of feature observations per frame



(a)



(b)

Fig. 6: Monte-Carlo study comparing between LBA and full SLAM, both using incremental smoothing: (a) Accelerometer bias estimation errors; (b) Gyroscope bias estimation errors. Best viewed in color.

γ, which affects the number of observed landmarks. We therefore discuss processing time for two different values of feature observations per frame, $\gamma = \{200, 500\}$, while performing exactly the same trajectory. In the former case, number of landmarks is 9.5k with total number of image observations of about 66k, while in the latter case, number of landmarks and total number of image observations are 23k and 165k, respectively.

Processing time for these two cases is shown in Figures 7c-7d and summarized in Table I. As seen, while BA exhibits lower processing time now and then, in particular when far from loop closure frames, the overall processing time is much smaller in the LBA case. One can clearly observe the spikes in BA, that are the result of massive variable re-elimination and re-linearization triggered by loop closures and proceeds for many frames afterwards. Overall, the average processing time per frame in the shown scenario for $\gamma = 200$ features is 0.27 and 0.59 seconds for LBA and BA, respectively. Increasing number feature observations per frame to $\gamma = 500$, leads to further difference in average processing time, as shown in Figure 7d: 0.57 and 1.95 seconds for LBA and BA. Thus, LBA is about 2 times faster, on average, than BA for $\gamma = 200$ features, and almost 5 times faster for $\gamma = 500$ features.

#Features per frame	#Landmarks	#Observations	Ave. Time [sec]		
			BA	**LBA**	Ratio
200	9.5k	66k	0.59	**0.27**	2.19
500	23k	165k	1.95	**0.57**	3.42

TABLE I: Average processing time per camera frame.

VII. Conclusions and Future Work

We presented a new computationally-efficient approach for robotics navigation in the challenging but common setup of a robot equipped only with an IMU and a single monocular camera. In order to incorporate incoming visual observations and, in particular, loop closure measurements, it is common to apply bundle adjustment optimization that involves both the navigation states and the observed 3D points. We instead algebraically eliminate the observed 3D points and formulate the cost function, using multi-view constraints, only in terms of navigation states, leading to a considerably reduced number of variables in the optimization. To avoid adding navigation states to the optimization at IMU rate, consecutive IMU measurements are summarized using a recently-developed technique. Applying incremental inference, only part of the navigation states are actually re-calculated for each new added measurement.

A statistical simulative study in which the proposed method was compared to full bundle adjustment was presented showing similar levels of accuracy in both methods for the considered scenario. Processing time was analyzed in a 13 km length simulated scenario, considering different number feature observations per frame. The analysis indicated that the proposed method, incremental light bundle adjustment, is $2 - 3.5$ time faster on average, compared to incremental bundle adjustment and this ratio increases with number of features per frame. Future work will focus on conducting large-scale experiments and developing a methodology for optimally adding new multi-view factors to the optimization.

Acknowledgments

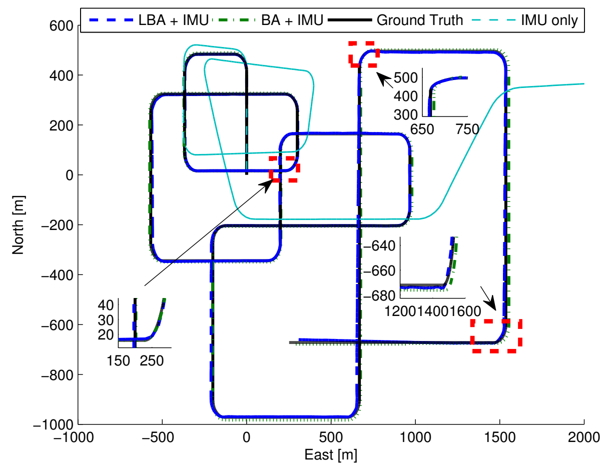
This work was partially supported by the DARPA All Source Positioning and Navigation (ASPN) program under USAF/ AFMC AFRL contract No. FA8650-11-C-7137 and by the ARL Micro Autonomous Systems and Technology (MAST) program under contract No. W911NF-08-2-0004. The authors would like to thank Dr. Stephen Williams, College of Computing, Georgia Institute of Technology, for insightful discussions.

References

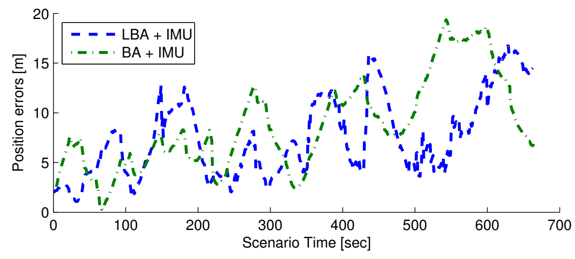
[1] S. Avidan and A. Shashua. Threading fundamental matrices. *IEEE Trans. Pattern Anal. Machine Intell.*, 23(1):73–77, 2001.

[2] 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.

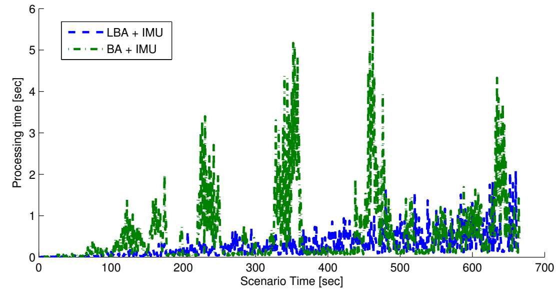
[3] J.A. Farrell. *Aided Navigation: GPS with High Rate Sensors*. McGraw-Hill, 2008.



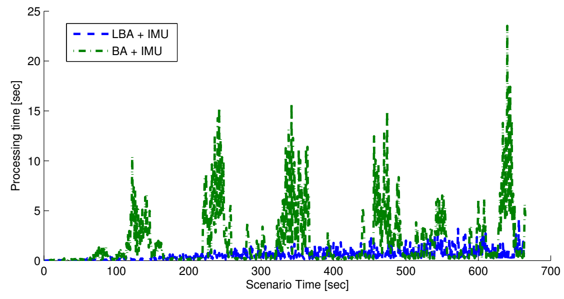
(a)



(b)



(c)



(d)

Fig. 7: (a) Top view of estimated trajectory for the scenario shown in Figure 1; (b) Position estimation errors (norm); (c) and (d) Processing timing comparison between the proposed method and bundle adjustment for 200 and 500 features per frame. Both methods use incremental smoothing and equivalent IMU factors.

[4] Strasdat H., Montiel J. M. M., and Davison A. J. Scale drift-aware large scale monocular SLAM. In *Robotics: Science and Systems (RSS)*, Zaragoza, Spain, June 2010.

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

[6] F.S. Hover, R.M. Eustice, A. Kim, B.J. Englot, H. Johannsson, M. Kaess, and J.J. Leonard. Advanced perception, navigation and planning for autonomous in-water ship hull inspection. *Intl. J. of Robotics Research*, 31(12):1445–1464, Oct 2012.

[7] V. Ila, J. M. Porta, and J. Andrade-Cetto. Information-based compact Pose SLAM. *IEEE Trans. Robotics*, 26(1), 2010. In press.

[8] V. Indelman. *Navigation Performance Enhancement Using Online Mosaicking*. PhD thesis, Technion - Israel Institute of Technology, 2011.

[9] 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.

[10] V. Indelman, R. Roberts, C. Beall, and F. Dellaert. Incremental light bundle adjustment. In *British Machine Vision Conf. (BMVC)*, September 2012.

[11] V. Indelman, R. Roberts, and F. Dellaert. Probabilistic analysis of incremental light bundle adjustment. In *IEEE Workshop on Robot Vision (WoRV)*, January 2013.

[12] 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.

[13] V. Indelman, S. Wiliams, M. Kaess, and F. Dellaert. Information fusion in navigation systems via factor graph based incremental smoothing. *Robotics and Autonomous Systems*, 61(8):721–738, August 2013.

[14] 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.

[15] 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.

[16] K. Konolige. Sparse sparse bundle adjustment. In *British Machine Vision Conf. (BMVC)*, September 2010.

[17] K. Konolige and M. Agrawal. FrameSLAM: from bundle adjustment to realtime visual mapping. *IEEE Trans. Robotics*, 24(5):1066–1077, 2008.

[18] 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.

[19] M.I. A. Lourakis and A.A. Argyros. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Trans. Math. Software*, 36(1):1–30, 2009.

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

[21] 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.

[22] 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.

[23] 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.

[24] P. Moutarlier and R. Chatila. An experimental system for incremental environment modelling by an autonomous mobile robot. In *Experimental Robotics I, The First International Symposium, Montréal, Canada, June 19-21, 1989*, pages 327–346, 1989.

[25] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in Robotics. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer-Verlag, 1990.

[26] S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *Intl. J. of Robotics Research*, 23(7-8):693–716, 2004.

[27] R. Vidal, Y. Ma, S. Soatto, and S. Sastry. Two-View Multibody Structure from Motion. *Intl. J. of Computer Vision*, 68(1):7–25, 2006.