Incremental Light Bundle Adjustment for Structure From Motion and Robotics

Vadim Indelman*, Richard Roberts‡, and Frank Dellaert‡

Abstract

Bundle adjustment (BA) is essential in many robotics and structure-from-motion applications. In robotics, often a bundle adjustment solution is desired to be available incrementally as new poses and 3D points are observed. Similarly in batch structure from motion, cameras are typically added incrementally to allow good initializations. Current incremental BA methods quickly become computationally expensive as more camera poses and 3D points are added into the optimization. In this paper we introduce incremental light bundle adjustment (iLBA), an efficient optimization framework that substantially reduces computational complexity compared to incremental bundle adjustment. First, the number of variables in the optimization is reduced by algebraic elimination of observed 3D points, leading to a structureless BA. The resulting cost function is formulated in terms of three-view constraints instead of re-projection errors and only the camera poses are optimized. Second, the optimization problem is represented using graphical models and incremental inference is applied, updating the solution using adaptive partial calculations each time a new camera is incorporated into the optimization. Typically, only a small fraction of the camera poses are recalculated in each optimization step. The 3D points, although not explicitly optimized, can be reconstructed based on the optimized camera poses at any time. We study probabilistic and computational aspects of iLBA and compare its accuracy against incremental BA and another recent structureless method using real-imagery and synthetic datasets. Results indicate iLBA is 2-10 times faster than incremental BA, depending on number of image observations per frame.

1 Introduction

Bundle adjustment (BA) has been at the focus of many research efforts for the past few decades. While its origin is in the photogrammetry community, it is an essential component in many applications in computer vision and robotics,

* Corresponding author, Department of Aerospace Engineering, Technion - Israel Institute of Technology, Haifa 32000, Israel. Email: vadim.indelman@technion.ac.il.

‡Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30332, USA.

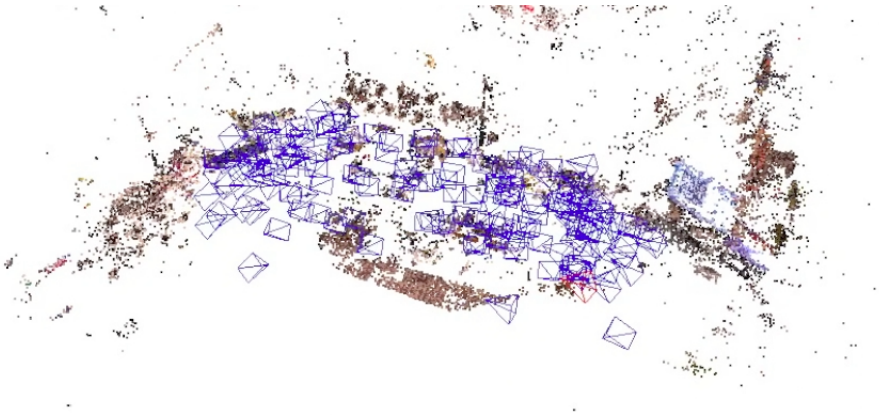
where it is also known as structure from motion (SfM) and full simultaneous localization and mapping (SLAM). A thorough review of different aspects in BA can be found in [49].

Bundle adjustment can be considered as a large optimization problem, with the optimized variables being the camera poses and the observed structure (3D points/landmarks) and where one aims to minimize the difference between the actual and the predicted image observations. Computational complexity is scenario-dependent and is a function of several factors, including the number of images, observed 3D points, and the actual image observations. Research in recent years has been focused on developing efficient and fast approaches for BA optimization, which is required to process a large amount of information in a reasonable time. Proposed approaches include exploiting the typical sparseness of the problem, performing parallelization and distribution of the involved calculations, and optimizing a reduced system that approximates the full system.

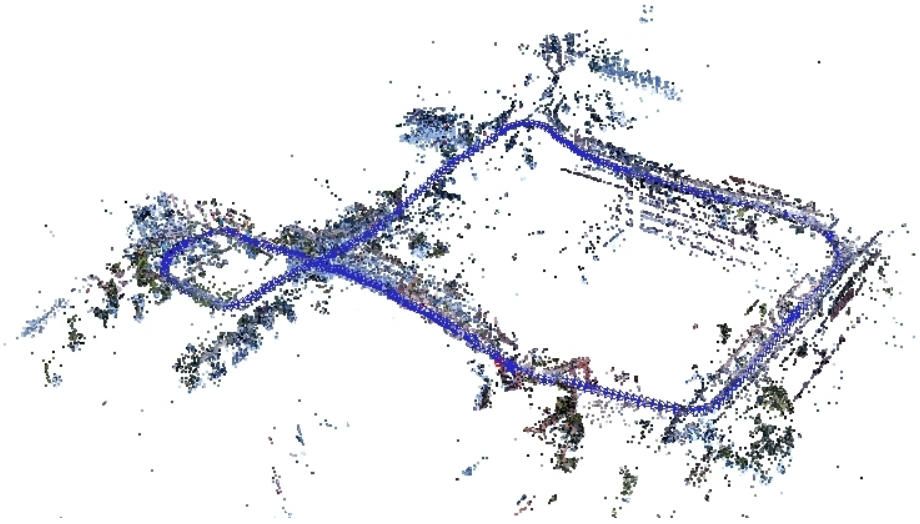
This paper introduces incremental light bundle adjustment (iLBA), an efficient optimization method that is applicable both to SfM and SLAM. The method incorporates two key components to reduce computational complexity: structureless BA and incremental smoothing.

The idea in the recently-developed structureless BA methods [41, 18, 21] is to improve computational complexity by reducing the number of variables involved in the optimization. Specifically, in structureless BA only the camera poses are optimized, while the 3D points are algebraically eliminated using multi-view constraints. The optimization minimizes the errors in satisfying these constraints, instead of optimizing the re-projection errors as in standard BA. iLBA utilizes three-view constraints [17], which in contrast to using only epipolar constraints in structureless BA [41], allow consistent motion estimates even when the camera centers are co-linear, a situation common in mobile robotics. Alternatively, one can apply trifocal constraints [35, 15] within the same framework. If required, all or some of the 3D points can be calculated using standard structure reconstruction techniques using the optimized camera poses. Figure 1 shows an example of iLBA-optimized camera poses, which are then used for sparse 3D reconstruction in two real-imagery datasets considered in this paper.

The second component in iLBA is incremental smoothing, an efficient incremental optimization that uses a recently-developed technique [27] to update the solution using adaptive partial calculations each time a new camera (image) is incorporated into the optimization. Being able to incorporate new information and efficiently perform the optimization is important in many applications, such as robot vision, where the images are not given ahead of time (e.g. gradually captured by a single or group of mobile robots) and the maximum a posteriori (MAP) estimate is required each time a new image is received. Furthermore, since a good initialization is essential not to be trapped in a local minima, incremental optimization is also used in batch scenarios where all the imagery data is available. Typically, incremental smoothing involves recalculating only a small number of camera poses as opposed to always optimizing all camera poses in previous incremental SfM [53] and structureless BA methods [47, 41, 18]. To the best of our knowledge, incremental smoothing has not been suggested to



(a) *Cubicle* dataset: 148 cameras, 31,910 3D points and 164,358 image observations.



(b) *Outdoor* dataset: 308 cameras, 74,070 3D points and 316,696 image observations.

Figure 1: Optimized camera poses using iLBA followed by (sparse) structure reconstruction. Computational time of iLBA is 8-10 times faster compared to incremental BA.

structureless BA thus far.

We demonstrate, using different real-imagery and synthetic datasets, that these two components result in a significant reduction in computational complexity while exhibiting high accuracy for reasonable image noise levels, compared to standard BA. Our implementation of iLBA is publicly available[1].

In this paper we also present a probabilistic analysis of iLBA, analyzing how well the probability distribution corresponding to the iLBA cost function agrees with the true probability distribution of the camera poses. The latter can be calculated in standard BA from the joint probability of camera poses and 3D points. An accurate and reliable uncertainty estimate is important in many structure from motion and robotics applications, yet to the best of our knowledge this is the first time that such an analysis is conducted for structureless BA methods.

Consequently, this paper makes the following main contributions: a) An efficient incremental structureless bundle adjustment method is introduced. b) Probabilistic analysis of thereof, as well as of previous structureless methods, is presented. c) Accuracy and computational complexity are evaluated and compared to other methods based on synthetic and real-imagery datasets.

The present paper is an extension of the work presented in [21, 22]. As a further contribution, in this manuscript we present an extensive quantitative evaluation of the performance and accuracy of iLBA, considering different aspects such as number of image observations per frame, different feature track lengths, dense sequential loop closures and sensitivity to image noise.

The remaining part of this paper is organized as follows. Related work and background material on standard and structureless BA are discussed in Sections 2 and 3, respectively. Light bundle adjustment (LBA), the first component in our approach, is introduced in Section 4. The second component is then discussed in Section 5, where LBA is reformulated for a recently-developed incremental factor graph optimization method, iSAM2 [27]. Computational complexity and evaluation of timing performance using synthetic and real-imagery datasets is provided in Section 6. Probabilistic analysis, and study of accuracy and uncertainty estimation are given in Section 7. Concluding remarks and directions for future research are suggested in Section 8.

2 Related Work

A vast body of work describes approaches for reducing the computational complexity of BA, in particular when processing a large number of images. Lourakis and Argyros [32] and Konolige [29] develop methods that fully exploit the sparsity of the involved matrices in the optimization. Snavely et al. [44] construct a skeletal graph using a small subset of images and incorporate the rest of the images using pose estimation, while Konolige and Agrawal [30] carefully choose

[1] Code is available on the website of the first author.

and optimize a reduced non-linear system, that approximates well the full system. Another family of methods used to reduce the complexity of optimizing the standard BA problem use preconditioned gradient descent [1, 4, 24, 25].

Increased efficiency and substantial improvement in running time have been obtained by using parallel and distributed computations. For example, Ni et al. [39] optimize in parallel several submaps that are merged afterwards, and Wu et al. [52] efficiently solve large 3D reconstruction problems by exploiting multicore CPU and GPU. GPU-based parallelization is also used by Frahm et al. [12], whose method can be applied both to large collections of images and to video streams. Klein and Murray [28] developed an approach for high rate parallel tracking and mapping using a monocular camera, while Newcombe et al. [38] presented an approach for real time camera tracking and dense reconstruction. Agarwal et al. [2] presented a 3D reconstruction system that exploits parallelization, using multi-core and cloud computing, to be able to handle large-scale image sequences.

Incremental bundle adjustment is a topic of recent research. Most existing work maintains real-time performance by optimizing a small subset of the most recent poses each time a new image is added. For example, Engels *et al.* [7] optimize a number of the most recent camera poses and landmarks using the standard BA optimization, and Mouragnon *et al.* [37] perform a similar optimization but including keyframe selection and using a generic camera model. Zhang and Shan [53] perform an optimization over the most recent camera triplets, experimenting with both standard BA optimization and a reduced optimization where the landmarks are linearly eliminated. However, since these methods optimize a sliding window of the most recent past poses, the necessary cameras will not be readjusted after large loop closures.

Other incremental bundle adjustment methods optimize more than just a fixed number of recent poses, instead adaptively identifying which camera poses to optimize [46, 42, 36]. However, in contrast with incremental smoothing [27], these methods result in approximate solutions to the overall least-squares problem, and do not reuse all possible computations. Additionally, these methods leverage specific properties of SfM problems, whereas incremental smoothing is a fully general incremental nonlinear optimization system, making possible immediate integration of additional robot-specific measurements such as GPS and IMU [23]. It is for these reasons incremental smoothing is our method of choice in this work.

The above methods efficiently minimize the overall re-projection errors, a procedure that involves optimizing both the camera poses and the observed 3D points (with the exception of Zhang and Shan). In contrast, structureless bundle adjustment approaches, which is also the approach in our work, algebraically eliminate the 3D points from the optimization and optimize the residual error in satisfying applicable multi-view constraints.

Early approaches suggested applying multi-view constraints for solving the SfM problem in a given sequence of images. For example, Fitzgibbon and Zisserman [11] suggested estimating the trifocal tensors [15] between consecutive triplets of images as well as the observed 3D points in each such triplet, fol-

lowed by registration of the triplets sequence and the involved structure into a single reference frame. Nistér [40] generalized this scheme by constructing a hierarchical structure of trifocal tensors, allowing to identify cameras with reliable estimated tensors and large parallax. The structure and motion problem is then solved using a standard bundle adjustment applied on these cameras and the associated feature correspondences.

The concept of avoiding structure estimation as an intermediate step has been proposed in several works. For example, Avidan and Shashua [3] used trifocal tensors to consistently concatenate sequential fundamental matrices, while Zhang and Shan [53] proposed local BA applied on a sliding window of triplets of images and correcting the image observations instead of estimating the 3D points that they represent. In [50], SfM is solved using a constrained least squares optimization, with the overall re-projection errors minimized subject to all independent two-view constraints.

While neither of the above methods have proposed a global optimization that is solely based on multi-view constraints, they paved the way to structureless BA where explicit 3D reconstruction is avoided altogether [47, 41, 18]. The first structureless BA method was introduced, to the best of our knowledge, by Steffen et al. [47], who optimized the corrections of image observations subject to satisfying trifocal tensor constraints. The authors used a relative parametrization of camera poses, with the pose of each camera expressed relative to the previous camera, resulting in improved numerical stability and faster convergence. A similar concept was developed in [18] using three-view constraints [19] instead of the trifocal tensor. While structure reconstruction is avoided, the number of variables in the optimization is larger than in case of a standard bundle adjustment, since in addition to the camera poses it also involves corrections to images observations. Rodríguez et al. [41] obtained a reduced computational complexity by reformulating the optimized cost function and refraining from correcting the image observations. The cost function in their formulation is expressed directly in terms of residual errors of two-view constraints.

Another related approach to SfM and SLAM is to optimize only the camera poses but using a graph of pairwise or triplet-wise relative pose constraints. In robotics, this is referred to as pose-SLAM, e.g. [9, 16], or FrameSLAM [30]. Similar approaches in SfM solve camera poses using graphs of pose constraints on camera pairs and triplets [13, 14]. The cost function in this approach is formulated using relative pose constraints, which are estimated in a separate process. In contrast, the cost function in structureless BA is expressed in terms of individual multi-view constraints.

3 Standard and Structureless Bundle Adjustment Methods

In this section we briefly explain two methods for bundle adjustment. Standard bundle adjustment optimizes both camera poses and 3D landmark positions

given image feature observations. Structureless bundle adjustment algebraically eliminates the 3D landmarks, thereby avoiding explicitly optimizing them.

3.1 Standard Bundle Adjustment

We consider a sequence of N images captured from different and unknown camera poses. Denote the camera pose that captured the ith image by x_i and let Z_i represent all the image observations of that image, with a single image observation of some landmark l_j denoted by $z_i^j \in Z_i$. Let X and L represent, respectively, all the camera poses and the observed landmarks,

$$X \doteq \{x_1, \ldots, x_N\} \quad , \quad L \doteq \{l_1, \ldots, l_M\}, \tag{1}$$

with M being the number of observed landmarks. These landmarks represent 3D scene points that generate the detected 2D visual features.

The joint probability distribution function (pdf) is then

$$p\left(X, L|\mathcal{Z}\right), \tag{2}$$

where $\mathcal{Z}$ is the set of all image observations from all images: $\mathcal{Z} \doteq \{Z_i\}_{i=1}^{N}$.

The maximum a posteriori (MAP) estimation of X and L is given by

$$X^*, L^* = \arg\max_{X,L} p\left(X, L|\mathcal{Z}\right). \tag{3}$$

The joint pdf $p\left(X, L|\mathcal{Z}\right)$ can always be factorized in terms of any available a priori information and the measurement models as

$$p\left(X, L|\mathcal{Z}\right) = priors \cdot \prod_{i}^{N} \prod_{j}^{M} p\left(z_i^j | x_i, l_j\right). \tag{4}$$

When assuming Gaussian distributions, each individual measurement term $p\left(z_i^j | x_i, l_j\right)$ can be written as

$$p\left(z_i^j | x_i, l_j\right) \propto \exp\left(-\frac{1}{2}\left\|z_i^j - proj\left(x_i, l_j\right)\right\|_{\Sigma}^2\right) \doteq f_{proj}, \tag{5}$$

where $proj\left(\cdot\right)$ is the projection function [15] 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. The $priors$ term can be written accordingly and is omitted, from now on, for conciseness.

Calculating the MAP estimate then becomes equivalent to minimizing the following non-linear least-squares cost function (omitting the $priors$ for clarity):

$$J_{BA}\left(X, L\right) = \sum_{i}^{N} \sum_{j}^{M} \left\|z_i^j - proj\left(x_i, l_j\right)\right\|_{\Sigma}^2, \tag{6}$$

a process known as bundle adjustment.

One can observe that optimizing (6) involves $6N + 3M$ variables and thus becomes computationally expensive as the number of views and observed landmarks increase.

This is particularly an issue if new images become available on a constant basis, such as in robot vision applications, and a solution is required at some frequency (e.g. each time a new image is received). Denoting by X_k the camera poses of all images up to the current time t_k, and by L_k all the observed landmarks in these images, we are interested in finding the MAP estimate

$$X_k^*, L_k^* = \arg\max_{X_k, L_k} p\left(X_k, L_k | \mathcal{Z}_k\right), \tag{7}$$

with $\mathcal{Z}_k$ representing all the image observations. Note that not all cameras observe all landmarks, making the bundle adjustment problem usually very sparse.

3.2 Structureless Bundle Adjustment

Different approximations to the BA cost function (6) have been proposed in previous structureless BA methods [47, 41, 18]. One alternative is to express the cost function in terms of corrections made to the image observations, subject to satisfying applicable multi-view constraints [47, 18] (again omitting *priors*):

$$J_{SLB}\left(X, P\right) = \sum_{i}^{N} \sum_{j}^{M} \left\| z_i^j - p_i^j \right\|_{\Sigma}^2 - 2\lambda^T h\left(X, P\right), \tag{8}$$

where $p_i^j \in P$ denotes the "fitted" image observation $z_i^j \in \mathcal{Z}$, and P represents all the fitted observations. λ is a vector of Lagrange multipliers and $h \doteq \begin{bmatrix} h_1 & \dots & h_{N_h} \end{bmatrix}^T$ represents all the multi-view constraints derived from feature correspondences in the given sequence of views, with N_h being the number of such constraints. Each such constraint h_i is an implicit nonlinear function of camera poses and image observations, involving different views that observe a mutual scene. For example, Rodríguez et al. use epipolar geometry constraints [41], Steffen et al. employ trifocal tensor constraints [47], while Indelman uses three-view constraints [18]. Since our approach is also based on three-view constraints, we defer a detailed exposition of these constraints to the next section. We denote the structureless BA approach (8) by SLB and use this method with the mentioned three-view constraints in the rest of this paper.

Observe that since image observations are also optimized and each 3D point is typically observed in several views, the number of variables in the SLB cost function (8) is much higher compared to the BA cost function (6).

4 Light Bundle Adjustment (LBA)

In this section we describe the first component in our approach, LBA. In order to introduce the details of LBA, we first explain it in a batch scenario. We then

proceed to the incremental framework in the context of robotics in Section 5.

LBA reduces the number of variables in the bundle adjustment optimization, as compared with standard bundle adjustment, by eliminating the landmarks and thus optimizing only the camera poses. Instead of a probabilistic variable elimination from the pdf, LBA uses an algebraic elimination relying on multiview constraints to introduce a new pdf $p_{LBA}(X|\mathcal{Z})$ over only the camera poses.

Key to the efficiency of LBA is that its density over poses $p_{LBA}(X|\mathcal{Z})$ is *not* equivalent to the exact marginalization of the landmarks $p(X|\mathcal{Z}) = \int_L p(X, L|\mathcal{Z})\, dL$. Performing the latter marginalization would require first iterative nonlinear optimization of the full bundle adjustment problem, including landmarks, to find the MAP estimate of both the camera poses and landmarks before applying a Gaussian approximation to compute the marginal. This nonlinear optimization of the full bundle adjustment problem is the expensive task that LBA avoids.

The LBA pdf $p_{LBA}(X|\mathcal{Z})$ derives from Gaussian noise on a measurement residual $h(x, z) \to \mathbb{R}$ derived from the multiview constraints (omitting *priors*),

$$p_{LBA}(X|\mathcal{Z}) \propto \prod_{i}^{N_h} \exp\left(-\frac{1}{2}\left\|h_i(\bar{X}_i, \bar{Z}_i)\right\|^2_{\Sigma_i}\right). \tag{9}$$

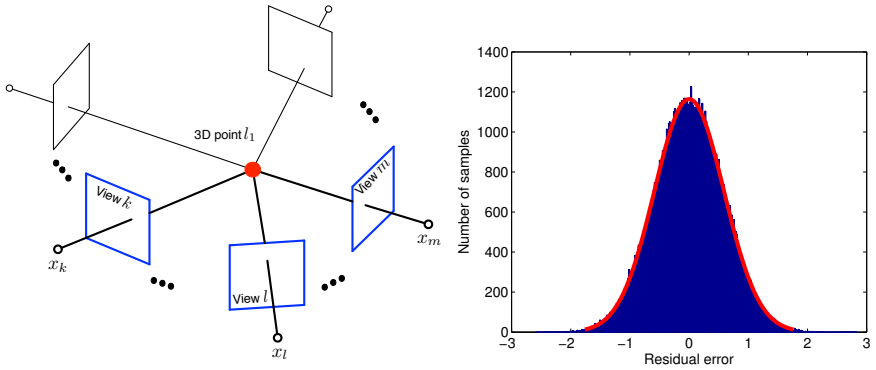
Here, i indexes over residual functions, and each $h_i\left(\bar{X}_i, \bar{Z}_i\right)$ is the residual of one of the multiview constraints in Eqs. (10)-(12), involving either two or three camera poses. We represent this ith subset of camera poses and the involved image observations by[2] $\bar{X}_i \subset X$ and $\bar{Z}_i \subset Z$, respectively.

Unlike in standard bundle adjustment, there is not a 1-1 mapping between residual functions and landmark observations. Instead, each residual function is the result of eliminating a landmark from the observations of two or three cameras. While a landmark may be observed by more than three views, algebraically independent relations exist only between up to three cameras [34]. Therefore, algebraic elimination of a landmark is represented by a set of two- and three-view constraints h_i between the different cameras observing this landmark. The procedure for selecting which residual functions are instantiated is described in Section 5.3.

The covariance matrices Σ_i are calculated as $\Sigma_i = A_i \Sigma A_i^T$, where A_i is the Jacobian of the constraint h_i with respect to the involved pixel observations. Although to yield the best accuracy Σ_i should be recalculated each time appropriate variables are re-linearized, in LBA we opt to calculate these matrices only once. We show in Section 7.3 that this has negligible effect on accuracy, while computational cost is significantly reduced.

As opposed to the SLB cost function (8), and similar to [41], image observations are not corrected in the LBA formulation.

[2]Not to be confused with X_i and Z_i.



(a) (b)

Figure 2: (a) Three view geometry for some three views k, l and m observing a 3D point l_1. (b) Histogram of the three-view constraint residual g_{3v}, showing a distribution that is very close to a Gaussian. The figure was generated by contaminating ideal image observations in three given views with sampled Gaussian noise (100000 samples). The standard deviation (std) of the fitted normal distribution (red curve) is very close to the calculated std $\sqrt{\Sigma_{3v}}$ (see Section 4). A similar distribution is obtained for the two-view constraint g_{2v}, as well as for different camera configurations.

4.1 Residual Functions

The LBA residual function is a proxy for the reprojection error of a landmark between two or three cameras. Consider the camera poses that observe some common 3D point l_1 as illustrated in Figure 2a. Writing down all the appropriate projection equations, it is possible to algebraically eliminate l_1, which results in constraints between triplets of poses [51]. One possible formulation of these constraints, recently developed in the context of vision-aided navigation [17, 19], is the three-view constraints. Assuming three overlapping views k, l and m, these constraints are

$$g_{2v}\left(x_k, x_l, z_k, z_l\right) = q_k \cdot \left(t_{k\to l} \times q_l\right) \tag{10}$$

$$g_{2v}\left(x_l, x_m, z_l, z_m\right) = q_l \cdot \left(t_{l\to m} \times q_m\right) \tag{11}$$

$$\begin{aligned} g_{3v}\left(x_k, x_l, x_m, z_k, z_l, z_m\right) = \\ \left(q_l \times q_k\right) \cdot \left(q_m \times t_{l\to m}\right) - \left(q_k \times t_{k\to l}\right) \cdot \left(q_m \times q_l\right) \end{aligned} \tag{12}$$

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 three views and facilitates maintaining a consistent scale for the translation vectors $t_{k\to l}$ and $t_{l\to m}$. These constraints were shown

to be necessary and sufficient conditions for observing the same 3D point by these three views [17]. In contrast to using only epipolar geometry constraints, use of three-view constraints (10)-(12) as well as the trifocal tensor, maintains a consistent scale even in co-linear camera configurations. The appendix provides further details regarding the relation of the three-view constraints to the standard trifocal tensor.

Referring to the LBA pdf (9), each constraint h_i is thus a single two- or three-view constraint ($h_i \in \{g_{2v}, g_{3v}\}$) that is a function of several camera poses and image observations in the corresponding images.

When a landmark is observed by more than three views, a single two-view (10) and three-view constraint (12) are added for each new view k and some earlier views l and m. The reason for not adding the second two-view constraint (between views l and m) is that this constraint was already used when processing these past views. In case a 3D point is observed by only two views, we add a single two-view constraint. We postpone the discussion on how the past views l and m are chosen until Section 5.3.

The LBA pdf (9) assumes a Gaussian distribution of each of the residuals (both g_{2v} and g_{3v}). To verify this is a reasonable assumption, ideal synthetic image observations across different views were contaminated with sampled Gaussian noise in different synthetic scenarios, followed by calculation of the residuals for two- and three-view constraints. The statistics of these residuals is indeed very close to Gaussian distribution, as shown in Figure 2b for the three-view constraints g_{3v}.

Remark: In this paper we assume a projective camera model with known calibration matrices K_i. However, if the latter is not accurately known, it can become part of the inference problem, in which case each of the constraints g_{2v} and g_{3v} will involve also the calibration matrices as variables. As a further possible extension, one can also consider writing the constraints without assuming a linear camera model, and instead, incorporate appropriate additional terms to account for non-linear effects such as radial distortion within the above definition of q_i [15]. Conceptually, this would allow applying the method directly on captured imagery, without first rectifying it. However, it remains to be seen if such an approach will accurately infer the parameters corresponding to the nonlinear model.

Data Association Data association is a necessary step of determining which image feature observations across several cameras are in fact observations of the same 3D point. Shared observations comprise the constraints that allow optimizing camera poses (and 3D landmark positions in the case of standard bundle adjustment). In this paper we do not address the data association problem, instead we decouple it from the bundle adjustment process as is typical in other work. For our experiments we use data associations provided by Bundler [43].

We note, however, that in the context of robot navigation, data association needs to be determined online. A common method to do so is using RANSAC [10] with an underlying fundamental matrix model [15]. Similarly to LBA (and other structureless BA approaches), this process also does not involve explicit reconstruction of 3D points. Of course, data association is often imperfect and outliers can be occasionally introduced into the inference process. One common approach to deal with outliers is to resort to M-estimators with an appropriate robust loss function (e.g. Cauchy function), see e.g. [48]. While not at the focus of this paper, we note that these techniques can be also applied to LBA formulation (9) by accordingly weighting the residual error of each constraint h_i. We leave further investigation of these aspects to future research.

4.2 Factor Graph Representation

At this point it is beneficial to represent a factorization of the joint pdf $p(X)$ over variables X using a graphical model, a *factor graph* [31], which will be later used for efficient incremental inference (see Section 5.1):

$$p(X) \propto \prod_i f_i(x_\alpha, x_\beta, \ldots). \tag{13}$$

Here, $f_i \to \mathbb{R}$ is a *factor* that involves some subset $\{x_\alpha, x_\beta, \ldots\}$ of the variables X. The membership of this subset of variables, called the *involved variables*, is predetermined for each factor f_i. These sets of involved variables comprise the sparsity of the function. The factors f_i represent measurement likelihoods or prior terms.

The factor graph is a bipartite graph with two type of nodes: variable nodes $x_\alpha \in X$ and factor nodes f_i. An edge $e_{\alpha i}$ between the factor node f_i and the variable node x_α exists if and only if the factor f_i involves the variable x_α.

The pdfs of both standard BA (4) and of LBA (9) are already given in the form (13). In particular, for LBA we define two- and three-view factors as follows:

$$f_{2v}(x_k, x_l) \doteq \exp\left(-\frac{1}{2}\left\|g_{2v}(x_k, x_l, z_k, z_l)\right\|^2_{\Sigma_{2v}}\right) \tag{14}$$

and

$$f_{3v}(x_k, x_l, x_m) \doteq \exp\left(-\frac{1}{2}\left\|g_{3v}(x_k, x_l, x_m, z_k, z_l, z_m)\right\|^2_{\Sigma_{3v}}\right), \tag{15}$$

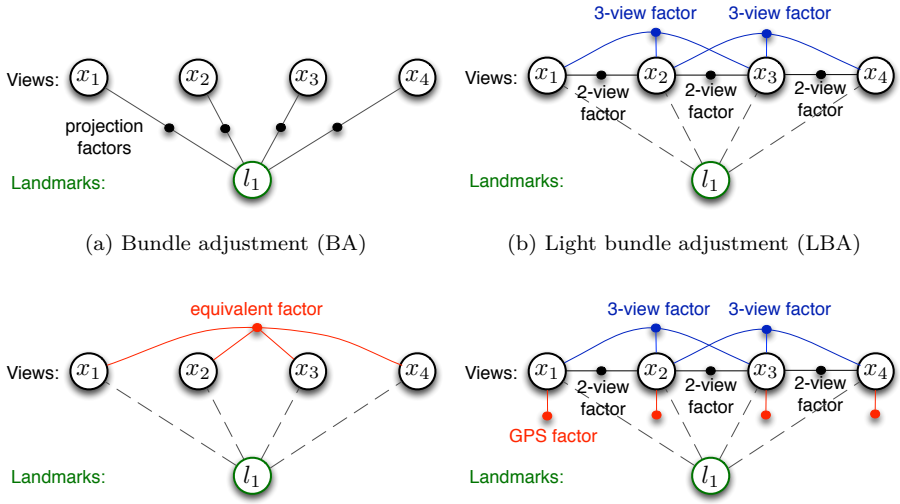
with the covariances Σ_{2v} and Σ_{3v}:

$$\Sigma_{2v} \doteq \left(\nabla_{z_k, z_l} g_{2v}\right) \Sigma \left(\nabla_{z_k, z_l} g_{2v}\right)^T \quad , \quad \Sigma_{3v} \doteq \left(\nabla_{z_k, z_l, z_m} g_{3v}\right) \Sigma \left(\nabla_{z_k, z_l, z_m} g_{3v}\right)^T. \tag{16}$$

Consequently, the LBA pdf (9) can be written as:

$$p_{LBA}(X|\mathcal{Z}) \propto \prod_{i=1}^{N_h} f_{2v/3v}\left(\bar{X}_i\right), \tag{17}$$

where, as before, $\bar{X}_i$ is a subset of camera poses.



(a) Bundle adjustment (BA)

(b) Light bundle adjustment (LBA)

(c) Marginalizing out a landmark

(d) LBA with GPS

Figure 3: Factor graph formulation for (a) BA and (b) LBA in a simple scenario of 4 cameras observing a single landmark l_1. (c) Factor graph after marginalizing out the landmark l_1. (d) LBA factor graph incorporating also GPS factors.

Similarly, the projection factor f_{proj} is defined in Eq. (5) and the BA pdf can be written accordingly. Figures 3a-3b illustrate factor graph representations of BA and LBA in a simple scenario of 4 cameras observing a single landmark.

4.3 LBA in Robotics

While so far we considered only visual observations, in actual robotic applications different additional sensors are typically available. Combining all the available measurements from different sensors then becomes a matter of information fusion. The joint pdf (17) will include, in addition to two- and three-view factors, also factors that represent measurement likelihood terms from these additional sensors [20].

For example, considering GPS measurements and assuming for simplicity the image and GPS measurements arrive at the same rate, the joint pdf can be written as (omitting *priors*)

$$p\left(X_k|\mathcal{Z}_k\right) \propto \prod_{s=0}^{k} \left(p\left(z_{GPS}|x_s\right) \prod_{i=0}^{n_s} f_{2v/3v}\left(\bar{X}_{s_i}\right) \right), \tag{18}$$

where the GPS basic measurement likelihood is represented by the unary factor $p\left(z_{GPS}|x_s\right)$, and n_s is the number of two- and three-view factors that are added

between camera pose x_s and past camera poses (i.e. $\bar{X}_{s_i} \subseteq X_s$). Figure 3d illustrates a factor graph corresponding to Eq. (18) in a simple scenario.

Given a factor graph representing a factorization of the joint pdf, one can calculate the MAP estimate X_k^* using the batch optimization described in Section 4.4. The same framework can also be used for inferring additional variable types, such as the robot's velocity and calibration parameters of appropriate sensors.

4.4 Batch Optimization

Minimizing the negative log-likelihood of the LBA pdf (17),

$$X^* = \arg\min_X -\log p_{LBA}\left(X|\mathcal{Z}\right), \tag{19}$$

or a general pdf $p\left(X|\mathcal{Z}\right)$ that includes additional sensors, leads to a nonlinear least-squares optimization over only the camera poses.

Gauss-Newton nonlinear optimization is suitable when the system is initialized sufficiently close to the solution. As with standard BA, the initialization must be of sufficient quality to not be trapped in local minima, though in practice we observe LBA to be more robust to bad initialization than standard BA. As opposed to BA, initialization of 3D points is not required in LBA. Additionally, standard BA typically requires a trust-region optimization like Dogleg or Levenberg-Marquardt for convergence [49], which adds computational complexity to the optimization. LBA, on the other hand, in all of our experiments, has converged using standard Gauss-Newton, without the use of trust-region methods.

Nonetheless, obtaining a good initialization prior to optimizing all camera poses is in fact quite difficult. Often in most SfM pipelines, instead of attempting to initialize all cameras at once, cameras or camera groups are added incrementally, with past cameras sometimes requiring nonlinear optimization before adding a new set [49]. LBA could also be used, instead of batch optimization each time a new camera is added, to speed up this procedure.

4.5 Structure Reconstruction

All, or some of the observed 3D points can be reconstructed based on the optimized camera poses following standard techniques (see, e.g., [15]). The optimal procedure, taking into account that both cameras and 3D points are random variables, is a full BA optimization (6). In practice, one may consider sub-optimal approaches that typically yield reasonable accuracy at lower computational cost. One sub-optimal approach is to reconstruct the jth 3D point l_j, observed in several views with indices represented by the set $\mathcal{A}_j$, by minimizing the following cost function

$$J\left(l_j\right) = \sum_{i \in \mathcal{A}_j} \left\| z_i^j - \mathcal{P}\left(x_i\right) l_j \right\|_\Sigma^2, \tag{20}$$

where $\mathcal{P}(x_i)$ represents the projection matrix calculated using the estimated pose x_i and the known calibration matrix K_i, so that $\mathcal{P}(x_i) l_j \equiv proj(x_i, l_j)$. While computational complexity of the optimization (20) depends on the cardinality of the set $\mathcal{A}_j$, it is typically inexpensive, in particular since it involves linear equations as camera poses are given and not optimized. The above formulation can be changed to include the camera pose uncertainty covariances and also account for correlation between different camera poses.

We stress again that structure reconstruction is *not* required by LBA, and should therefore be performed only when requested externally (e.g. by the user). It is for this reason that in the reported timing results for LBA (see Section 6), structure reconstruction is performed only once, after finishing processing the entire dataset. As discussed in the sequel, these results indicate LBA, including structure reconstruction, is significantly faster compared to BA.

5 Incremental Light Bundle Adjustment (iLBA)

In this section we describe the second component of our approach, extending LBA to the context of incremental inference in robotics. In robotics, information from cameras and other sensors typically comes in incrementally, and the best estimate incorporating all this available information should often be calculated as fast as possible. The main focus in this section is on LBA, and incorporating additional sensors is handled in the very same framework [23]. In order to efficiently handle the optimization component of iLBA, we use a recently-developed incremental smoothing method, iSAM2, which we review next.

5.1 Incremental Inference Using iSAM2

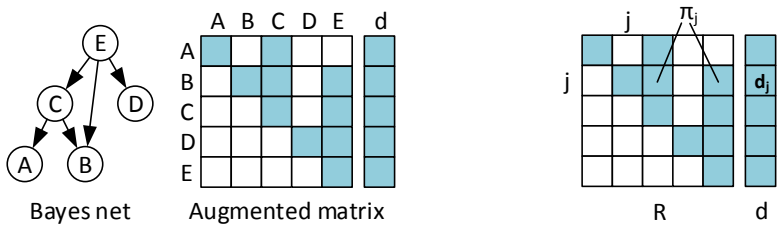
Our goal is to efficiently calculate the MAP estimate of the LBA pdf at time t_{k+1}, i.e. optimize all camera poses X_{k+1} that have appeared by time t_{k+1}, given all measurements $\mathcal{Z}_{k+1}$ that have arrived by that time:

$$X^*_{k+1} = \arg\min_{X_{k+1}} -\log p_{LBA}(X_{k+1}|\mathcal{Z}_{k+1}). \tag{21}$$

For this we use iSAM2, a recently-developed algorithm for very efficient incremental nonlinear optimization that reuses information obtained from optimizing the camera poses in the previous time step t_k.

Since understanding how iSAM2 operates is essential for understanding computational complexity, we now briefly discuss the basic concept of iSAM2. iSAM2 applies graph algorithms to a factor graph representation of the optimization problem to incrementally update an exact QR or Cholesky factorization of a linear approximation of the problem, while also fluidly relinearizing variables that move too far from their linearization points.

When a new image is obtained, each image correspondence between that image and previous images will generate new factors in the graph. This is



(a) Bayes net and corresponding augmented matrix.

(b) Example matrix entries corresponding to a conditional Gaussian.

Figure 4: Correspondence between a simple Gaussian Bayes net and the augmented matrix $[\,R\,d\,]$, obtained by variable elimination (i.e. QR or Cholesky decomposition). Hatched matrix entries are non-zero.

equivalent to adding new block-rows to the measurement Jacobian A 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.

iSAM2 works by incrementally updating a Gaussian Bayes net, equivalent to the upper-triangular matrix R resulting from QR or Cholesky decomposition of A – this decomposition is the main task in calculating the linear solution vector step δx in each iteration of the Gauss-Newton algorithm. The equivalency of a Gaussian Bayes net with sparse matrices is shown in Figure 4 and is as follows:

- A Gaussian Bayes net is the result of variable elimination on a Gaussian factor graph. Each node in the Bayes net represents a row of the sparse upper-triangular matrix R and the entry in the r.h.s. vector d with the same index as that row.

- Probabilistically, a Bayes net node, just as a row of the augmented matrix $[\,R\,d\,]$, represents the conditional Gaussian

$$p\left(x_j|\Pi_j\right) \propto \exp\left(-\frac{1}{2}\left\|R_{j,j}x_j + \left(\sum_{k\in\pi_j} R_{j,k}x_k\right) - d_j\right\|^2\right), \qquad (22)$$

  where j is the variable index (i.e. row index), $R_{\cdot,\cdot}$ is an entry of the matrix R, and d_j is the corresponding r.h.s. entry (see illustration in Figure 4b). Π_j is the set of immediate ancestor variables of x_j in the Bayes net, and π_j are the variable indices of these ancestors, equivalently the variable indices (i.e. column indices) with non-zero entries in row j of R.

iSAM2 proceeds to efficiently update the Bayes net with new variables and factors by leveraging the calculation dependencies represented by the Bayes net

structure to identify the set of variables that need to be recalculated or relinearized and then performing the elimination. We will often refer to this operation as *re-elimination*. A simplified version of the algorithm to determine the set of variables to be recalculated is listed in Algorithm 1. A formal exposition of the iSAM2 algorithm is given by Kaess *et al.* [27].

Algorithm 1 Identification of variables $x_{affected}$ to be recalculated in the Bayes net

Input: Bayes net, new factors, new variables
Initialization: $x_{affected} = \emptyset$
Locate all the variable nodes that are involved in the new factors *or* whose linear step δx is larger than a predetermined threshold.
for each such node v **do**
  Add v to $x_{affected}$
  Locate all paths in the Bayes net that lead from the node eliminated last in the elimination ordering (the root) to the node v
  for each such path **do**
    if a node v' is on the path and $v' \notin x_{affected}$ **then**
      Add v' to $x_{affected}$
    end if
  end for
end for

Example As a basic example, consider a factor graph that represents the LBA pdf with 3 and 4 camera poses, as shown in Figure 5. The former case (Figure 5a) includes only two-view factors, while in the latter case, there is also a three-view factor that relates between the forth, third and second camera poses. The corresponding Bayes net to the factor graph from Figure 5a, assuming variable ordering $\{x_1, x_2, x_3\}$, is illustrated in Figure 6a. This Bayes net represents[3] the factorization

$$p_{LBA}\left(X_3|\mathcal{Z}_3\right) = p_{LBA}\left(x_3|\mathcal{Z}_3\right) p_{LBA}\left(x_2|x_3,\mathcal{Z}_3\right) p_{LBA}\left(x_1|x_2,\mathcal{Z}_3\right). \tag{23}$$

Updating this factorization with a new camera pose (x_4) and two new factors, shown in red in Figure 5b, requires recalculating only some of the variables in the Bayes net. These variables, denoted by red color in Figure 6b, can be identified by Algorithm 1. One can observe the resulting Bayes net represents the factorization

$$p_{LBA}\left(X_4|\mathcal{Z}_4\right) = p_{LBA}\left(x_4|\mathcal{Z}_4\right) p_{LBA}\left(x_3|x_4,\mathcal{Z}_4\right) p_{LBA}\left(x_2|x_3,x_4,\mathcal{Z}_4\right) p_{LBA}\left(x_1|x_2,\mathcal{Z}_4\right) \tag{24}$$

where the term $p_{LBA}\left(x_1|x_2,\mathcal{Z}_4\right)$ is exactly the same as in the previous step (i.e. $p_{LBA}\left(x_1|x_2,\mathcal{Z}_4\right) \equiv p_{LBA}\left(x_1|x_2,\mathcal{Z}_3\right)$) and can therefore be re-used.

We proceed by addressing additional aspects in the iLBA framework.

[3] Explicit conditioning on the measurements is omitted from Figure 6.

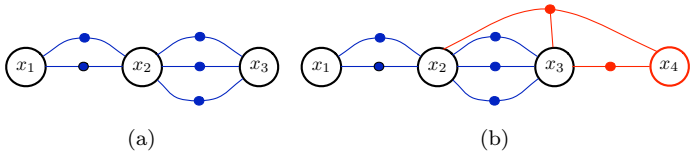


Figure 5: Example factor graph representation of (a) $p_{LBA}(X_3|\mathcal{Z}_3)$ and (b) $p_{LBA}(X_4|\mathcal{Z}_4)$. Red color denotes a new camera pose and factors added at time t_4.

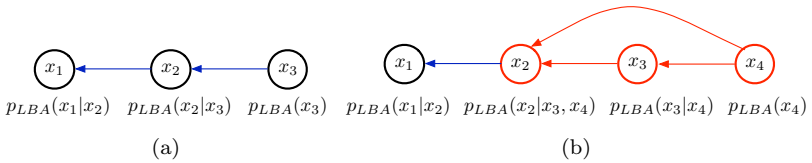


Figure 6: Incremental Bayes net update. (a) Corresponding Bayes net to the Factor graph from Figure 5a; (b) Corresponding Bayes net to the Factor graph from Figure 5b. Red color denotes re-calculated nodes, without taking re-linearization into account. In practice, typically only a small part of the Bayes net is re-calculated.

5.2 Incremental Initialization of New Cameras

Consider all camera poses up to time t_k have been already optimized, and a new image is obtained at time t_{k+1}. The optimization requires an initial value for the new camera pose x_{k+1}, and new two- and three-view factors between that pose and earlier camera poses.

Calculating an initial pose of x_{k+1} given some previously-processed two overlapping images involves two steps: we first calculate the relative motion between the new image and one of these previous images. This can be done by using the two-view constraints, or by calculating an essential matrix and extracting the motion parameters from it [15]. Since the translation is known only up to a scale, we optimize the translation magnitude using the three-view constraint (12) on camera triplets while keeping the rest of the motion parameters fixed.

5.3 Incrementally Selecting LBA Multiview Constraints Camera Sets

Next, new two- and three-view factors are created. For each new image observation $z_{k+1}^j \in Z_{k+1}$ of some landmark j, we locate two earlier camera poses l and

m that observe the same landmark (i.e. $z_l^j \in Z_l$ and $z_m^j \in Z_m$) and formulate the appropriate two- and three-view factors $f_{2v}\left(x_{k+1}, x_l\right)$ and $f_{3v}\left(x_{k+1}, x_l, x_m\right)$, as explained in Section 4.2. As opposed to BA, where a projection factor would be added between two specific variable nodes (new camera pose x_{k+1} and landmark node l_j), in LBA the past cameras l and m are not uniquely determined. Indeed, if the set $\mathcal{D}_k^j$ represents all the camera poses up to time t_k that observe the landmark l_j

$$\mathcal{D}_k^j \doteq \left\{ x_i | z_i^j \in Z_i \right\}, \tag{25}$$

then any two camera poses in $\mathcal{D}_k^j$ can be used to formulate the two factors above[4].

The choice of the past views l and m affects the graph topology and hence the computational complexity. It also affects accuracy which depends on the geometry between the camera poses, as well as on the number and distribution of image matches. While a rigorous analysis of how to optimally set the past views is left for future research, we have tried several different methodologies and use the following heuristic in this paper: we set m to be the earliest view in $\mathcal{D}_k^j$, while l is chosen such that the relative translation vectors $t_{k+1\to l}$ and $t_{l\to m}$ have similar magnitudes. Specifically, in our current implementation, we initialize l to $m + \lfloor (k-m)/2 \rfloor$ and gradually modify it while comparing between $|t_{k+1\to l}|$ and $|t_{l\to m}|$. Typically, very few attempts are required until an appropriate view l is found. However, if no such view l exists in $\mathcal{D}_k^j$, then the image observation z_{k+1}^j is skipped and no factors are created.

After creating new factors for all image observations Z_{k+1} in the current image, incremental inference is performed to calculate the MAP estimate (19), as explained in Section 5.1.

6 Computational Complexity Analysis

In this section we investigate the computational complexity of incremental LBA and compare it to incremental BA and incremental SLB in synthetic and real-imagery datasets.

Computational complexity depends both on the cost for solving the linearized system $A\Delta = b$, and on the number of performed iterations, in which appropriate variables are re-linearized. The LBA and BA Jacobians differ in size and in the number of non-zero elements, with LBA Jacobian having up to twice the number of factors but many fewer variables, since landmark variables are algebraically eliminated beforehand.

Computational complexity of incremental LBA depends on different aspects: graph topology, variable ordering, number of affected variables and the factors

[4]Moreover, the two- and three-view factors do not necessarily have to share the past view l, and can be defined instead using different past views l, l', m, i.e. $f_{2v}\left(x_{k+1}, x_l\right)$ and $f_{3v}\left(x_{k+1}, x_{l'}, x_m\right)$.

that involve these variables, number of iterations, and linearization thresholds of each variable type.

While computational complexity strongly depends on variable ordering, calculating an optimal ordering is NP hard. Thus an ordering is typically determined using approximate heuristics such as COLAMD [5], and the resulting complexity is therefore difficult to predict. Moreover, variables' re-linearization depends on thresholds being used for each variable type and analyzing how many variables are going to be re-linearized ahead of time is not trivial. For these reasons, in this section we analyze computational complexity of incremental LBA, compared to incremental BA, in an empirical study over a typical synthetic scenario, while assuming the ordering is calculated by constrained COLAMD in both cases. Additionally, we compare timing performance of incremental LBA, BA and another structureless BA method (SLB, see Section 3.2) in two real-imagery datasets.

Unless specified otherwise, all the methods in this study and in the remaining part of the paper are optimized incrementally using iSAM2; we therefore omit occasionally the explicit notation for incremental optimization (e.g., LBA refers to incremental LBA).

Before presenting this study, it is useful to point out the relation between the exact distribution over camera poses $p\left(X|\mathcal{Z}\right)$ and LBA distribution $p_{LBA}\left(X|\mathcal{Z}\right)$, in the context of computational complexity. The exact distribution can be obtained by marginalizing out all the landmark variables, as in Eq. (26), thereby assuming variable elimination ordering in which these variables are eliminated first, followed by the camera pose variables. Such an ordering has been commonly used in SfM related problems in computer vision (e.g. partitioned sparse LM optimization [15, 49]). Considering this specific ordering, LBA is guaranteed to have less fill-in since some of the inter-camera links are dropped (as explained in Section 7.1), and therefore both elimination and back-substitution steps will be faster.

In the next section, we consider the effect of the following aspects on computational complexity of both LBA and BA: number of image observations per frame, different feature track lengths, and dense sequential loop closures.

6.1 Complexity Study using Synthetic Datasets

We study different aspects in computational complexity of incremental LBA using two datasets, shown in Figure 7, in which the camera points downwards and observes 3D points while exploring different regions and occasionally revisiting previously-observed areas. The first dataset consists of 450 views and 15000 3D points; the second dataset consists of 350 views and 16500 3D points. While in these two datasets the camera points downwards, we consider also the other common scenario of a forward looking camera and present experiment results for two different scenarios in Section 7.4.

Each new camera pose is initialized based on the optimized pose of the previous camera and the relative pose obtained from an estimated essential matrix.

A perfect and known data association is assumed. Relative translation scale is determined using three-view geometry, as explained in Section 5.2. Landmarks, in case of BA, were initialized using triangulation after two observations have been made. The reason for using only two observations to initialize landmarks is to ensure BA and LBA use the same available information at each time instant.

An important aspect that has a major effect on processing time is how to choose linearization thresholds for each variable type (cf. Section 5.1). These thresholds were chosen such that nearly identical accuracy[5] is obtained compared to incremental batch optimization, for each of the examined methods.

All results reported in this paper were obtained on a 2.2 GHz Core i7 laptop using a single-threaded implementation. iLBA and other methods used for comparison were implemented using the GTSAM factor graph optimization library[6] [6, 27]. C++ code of iLBA is publicly available on the website of the first author.

Number of image observations per frame One reason that LBA is often faster than standard BA is that upon adding a new camera to the standard BA system, each observed landmark variable must be re-eliminated. Thus, a *lower* bound on the number of variables to be re-eliminated in BA is the number of image observations η in the current image. This bound is not tight – the actual number of re-eliminated variables is always higher: at least some of the currently-observed η landmarks will be ordered further away from the last-eliminated variable (which is typically the previous camera pose if using, e.g. constrained COLAMD [5]), and all variables on the path from this last-eliminated variable to each of the observed landmarks will also be re-eliminated[7].

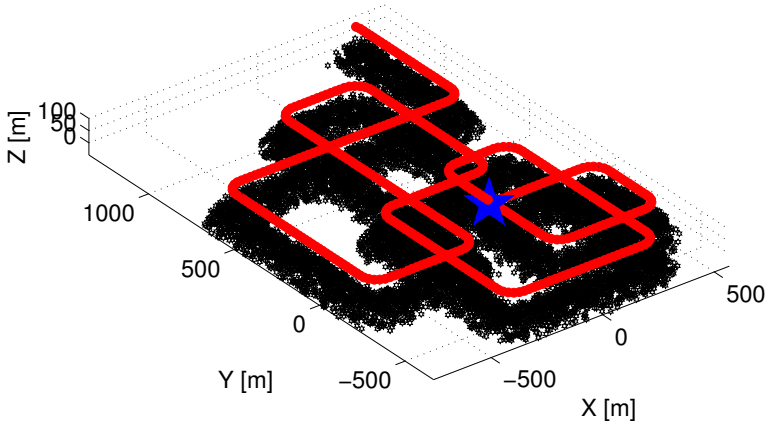
In contrast, the number of affected variables in LBA is *not* a function of η. Instead, this number depends on the variable ordering and on the method used for choosing past camera poses when adding new two- and three-view factors to the graph (cf. Section 5.2).

When adding new observations in iSAM2, in addition to re-eliminating the observed variables, the Bayes net ancestors of the observed variables must also be re-eliminated (see Algorithm 1). Therefore, the overall number of re-eliminated variables depends on the original elimination ordering. We can get further insight by examining a basic scenario without loop closures where the Bayes net structure is predictable. We consider the LBA factor creation heuristic used in this paper, according to which, given a new landmark observation, one of the past cameras involved in the generated three-view factor is chosen to be the earliest camera observing that landmark. If a sequential variable ordering is used, the number of re-eliminated variables is upper-bounded by $k - \zeta$,

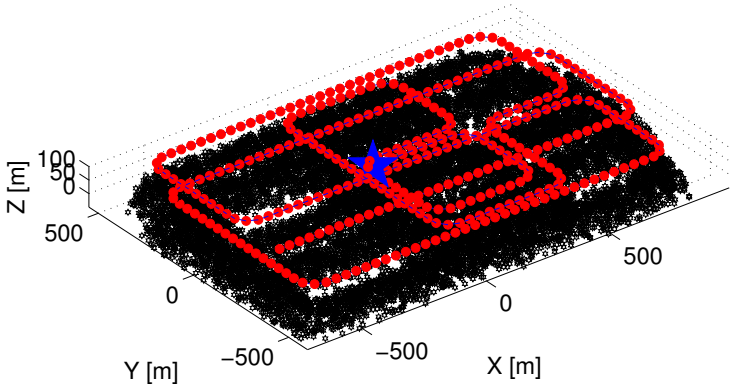
[5]The thresholds were set to the loosest values that result in maximum difference of 0.05 degrees in rotation and 0.5 meters in position and landmarks in the entire dataset, when comparing with batch optimization.

[6]https://borg.cc.gatech.edu/download.

[7]As mentioned in Section 5.1, the total number of variables re-eliminated also depends on linearization thresholds and on the number of required iterations until convergence.



(a)



(b)

Figure 7: Synthetic scenarios considered in Sections 6 and 7.3: Exploration with (a) occasional loop closures and (b) dense loop closures. Camera points downwards, and is shown in red, observed 3D points are shown in black. Beginning of scenario is denoted by a (blue) mark. Best viewed in color.

where k is the index of the new camera pose and ζ is the index of the earliest camera observing any of the η landmarks. If no loop-closures are present, i.e. all the η landmarks are consecutively observed, then $k - \zeta$ is the maximum feature track length. Note that this bound is *not* a function of the number of image observations per frame η. As will be seen next, although the actual variable ordering is not necessarily sequential, the number of variables re-eliminated in LBA, without taking into account re-linearization and loop closures, remains more or less the same for any value of η.

In more realistic and complex scenarios, the number of re-eliminated variables due to the LBA factor creation heuristic and the elimination order is best studied empirically. The effect of different number of image observations per frame on computational complexity both in LBA and BA is examined in Figure 8 for the above-mentioned synthetic scenario. Constrained COLAMD, standard in iSAM2, was used in both methods for calculating the variable ordering. As seen in Figure 8c, increasing η results in an increase on the lower-bound on variables re-eliminated in BA. Here, *lower-bounds* refer to the number of variables to be re-eliminated in a single iteration, as determined by graph topology and variable ordering and without considering re-linearization[8]. The number of variables re-eliminated in LBA, on the other hand, remains roughly the same as long as no loop closures occur and is very small (about 10 variables), *regardless* of the number of image observations per frame. The actual number of re-eliminated variables, taking into account all the performed iterations and also re-linearization of variables, is shown in Figure 8d.

During loop closures, the number of variables re-eliminated increases both in BA and LBA. One can clearly observe the spikes of variables re-eliminated in BA, while in LBA, processing loop closures involves re-eliminating many past camera variables, sometimes requiring re-eliminating all the variables. See lower bounds for these numbers in Figure 8e for $\eta = 200$. While these numbers are much smaller than in BA, this is an undesired behavior which results from the heuristic currently used for choosing what past camera poses to use when adding new factors (see Section 5.3). Investigating methodologies to address this issue is left for future research.

The number of factors that involve the re-eliminated variables, which is determined by graph topology, is another component that significantly affects computational complexity. Figure 8f shows the minimum number of factors (lower bounds) for $\eta = 200$, for LBA and BA. While away from loop closures, the number of involved factors is similar in both methods. However, since the number of re-eliminated variables is larger in BA, computational cost in LBA is much lower. The number of involved factors in LBA drastically increase during loop closures, which corresponds to the increase in re-eliminated variables as discussed above.

Figures 8a and 8b summarize processing time and number of variables re-eliminated for different values of η. As seen, lower bounds and the actual number

[8]Calculation of lower-bounds is performed using ideal measurements and perfect variable initialization so that indeed no re-linearization will be required and the number of variables to be re-eliminated is determined only by graph topology and variable ordering.

of variables re-eliminated are significantly smaller in LBA (Figure 8b).

An average processing time per frame, taking into account also re-linearization and including loop closures, is shown in Figure 8a. One can observe from this figure that LBA processing time is smaller by a factor of 2-3 in all cases. This speedup factor, i.e. the ratio between the shown two curves, increases with η, implying that LBA becomes more attractive (in terms of processing time) than BA as more image observations per frame are considered. For example, for $\eta = 400$, the speedup factor is around 2.5. As we show in the sequel (Section 7.3.2), higher values of η, i.e. using more image observations per frame, leads to better estimation accuracy.

We conclude by noting that in experiments we have observed LBA to be 8-10 times faster than BA (Section 7.4) due to higher values of η being used (on the order of 1000), as determined by extracting and matching SIFT [33] features.

Different feature track lengths In this section the effect of different feature track lengths on computational complexity of LBA and BA is investigated. Using the same dataset (Figure 7a), the typical feature track length was altered by changing the baseline between cameras: Gradually decreasing the baseline increases the typical feature track length. The same area is observed in all cases, resulting in more camera frames for longer feature track values. A fixed number of 200 image observations per frame is used.

Longer feature track length leads to more variables re-eliminated both in LBA and BA. In LBA, this effect is barely noticeable as long there are no loop closures: factors involve more distant pose nodes as typical feature track length increases. Typically, adding a new multi-view factor requires re-elimination of these pose variables. In contrast, we have observed that in BA the number of variables re-eliminated substantially increases with feature track length: Depending on the variable ordering, observation of a previously seen landmark involves more past cameras, and in case of BA also additional landmarks that are observed from these cameras.

This can be seen in Figure 9b, which shows the lower bound on number of variables re-eliminated (as determined by graph topology and the calculated ordering) for feature track lengths of 5, 15 and 30 camera frames. In practice, the actual number of variables re-eliminated is higher, as explained in the previous section.

Actual average processing time per frame is shown in Figure 9a. One can observe that LBA remains faster than BA also when feature track length increases. The two methods share the same trend of increasing computational time for longer feature track lengths, as these densify the graph, that contains larger cliques, and lead to more camera poses involved in each loop closure.

Dense sequential loop closures The effect of sequential dense loop closures on computational complexity is examined. The considered dataset, shown in Figure 7b, contains segments in which consecutive views introduce loop closure observations, i.e. the camera continuously re-visits previously observed areas.

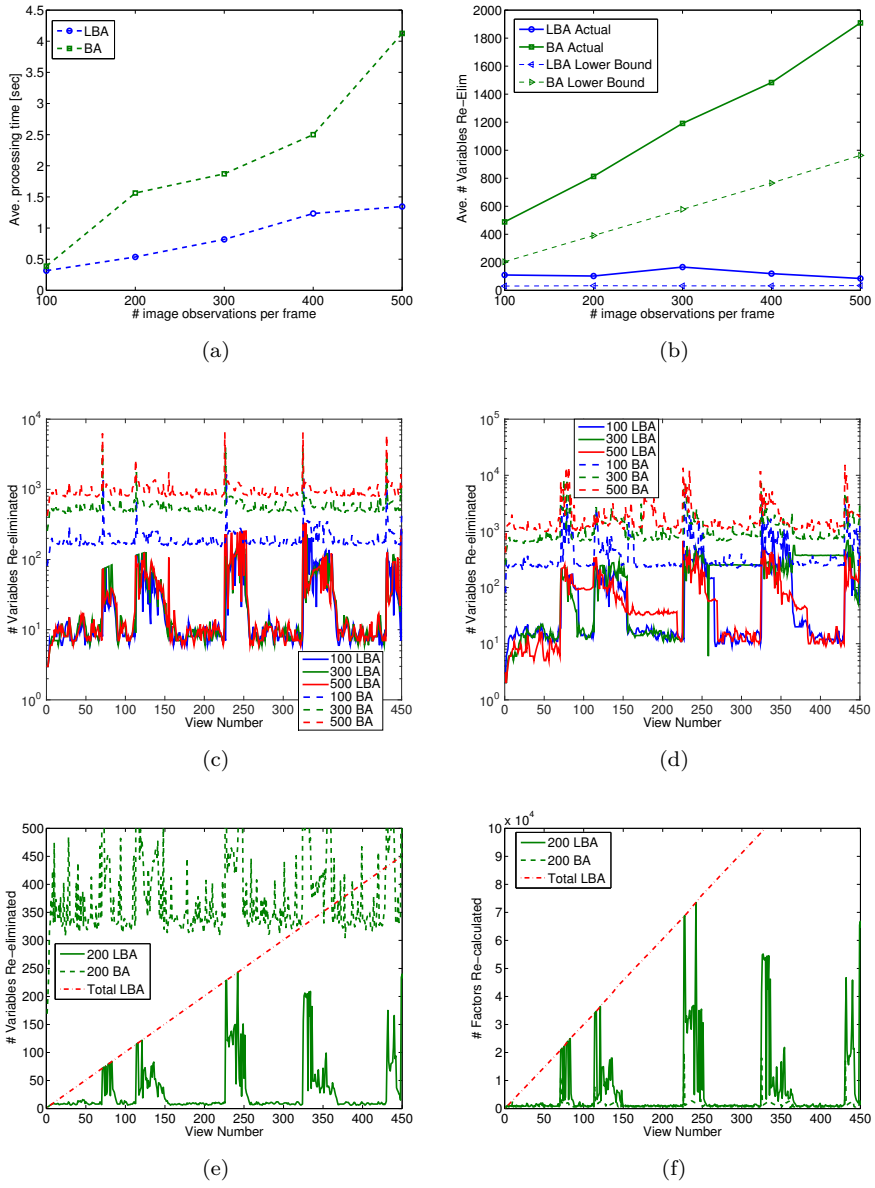
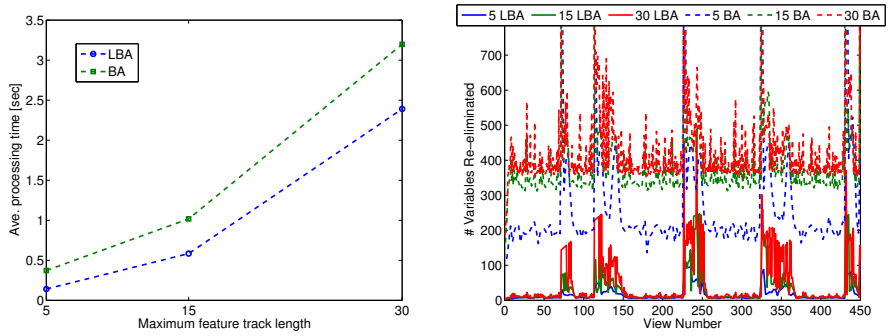


Figure 8: Effect of changing number of image observations per frame on computational complexity. Processing time of LBA is less than BA, and is less sensitive to the number of image observations per frame being used: (a) Average processing time per image. (b) Average number of re-eliminated variables per image. (c)-(d) Lower bounds and actual number of variables re-eliminated, per camera frame for $\eta \in \{100, 300, 500\}$. Loop closures are seen as spikes in the BA method. (e)-(f) Lower bounds for number of re-eliminated variables and the involved factors for $\eta = 200$ image observations per frame. Total number of variables and factors in LBA are also shown. Note scale in figures (c)-(d) is logarithmic.



(a) (b)

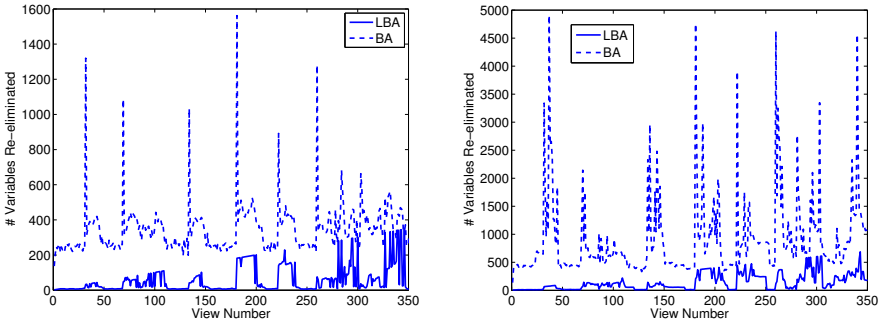
Figure 9: Effect of different feature track lengths (5, 15 and 30) on computational complexity aspects. Number of image observations per frame is $\eta = 200$ in all cases. (a) Average processing time per image. (b) Lower bounds on number of re-eliminated variables, determined by graph topology and variable ordering (corresponds to a single optimization iteration, without accounting for re-linearization). Loop closures can be clearly seen as spikes in the BA method (spikes go well beyond the shown y axis).

This mimics common SfM problems where many cameras view the same building or scene.

Lower bounds on the number of variables re-eliminated are shown in Figure 10a. Using the current heuristic for choosing past cameras when adding new factors, and calculating ordering based on constrained COLAMD, sequential loop closures involve re-eliminating many past cameras in LBA. Observe that in some cases all the camera poses are re-eliminated, as was already seen in the previous dataset. One can see the trend in the figure, according to which adding more cameras and loop closures will eventually result in BA having smaller lower bounds on number of variables re-eliminated than LBA.

However, the identified lower bounds are loose; the *actual* number of variables re-eliminated in LBA, taking into account affected variables due to re-linearization and the required iterations until convergence, is still much more attractive in most cases, as shown in Figure 10b. The corresponding average processing time per frame and total processing time is 0.59 and 210 seconds in LBA, compared to 0.75 and 268.5 seconds in BA.

Is there a better method for choosing past cameras in LBA? Is constrained COLAMD well-suited for LBA? These aspects are expected to impact both accuracy and computational complexity of LBA, and are left for future research. Further computational gain can be obtained by calculating a reduced measurement matrix [15] from the applied multi-view constraints, as was already presented in [41] for two-view constraints and can be extended to three-view constraints as well. Such an approach could remove the dependency of involved matrices



(a) (b)

Figure 10: Number of variables re-eliminated in presence of sequential dense loop closures. Number of image observations per frame is $\eta = 200$ in all cases. (a) Lower bounds; (b) Actual. Note the different scale in y axis in both graphs.

Dataset	Cameras	Landmarks	Observations
Cubicle	148	31910	164358
Outdoor	308	74070	316696

Table 1: Dataset details.

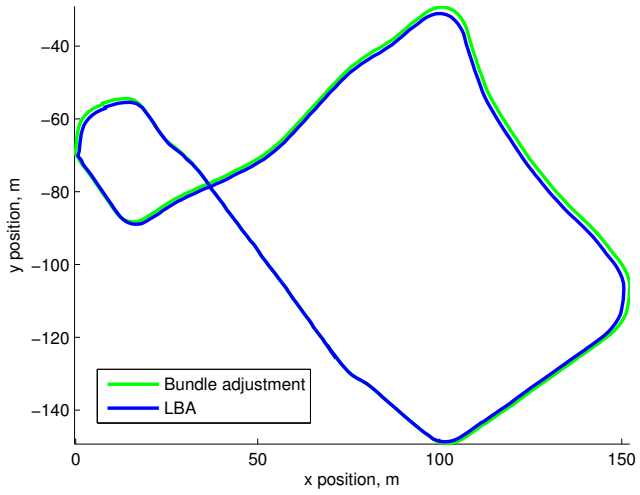
in the underlying optimization on the number of feature correspondences (or multi-view constraints), resulting in significantly reduced memory footprint.

6.2 Complexity Study using Real-Imagery Datasets

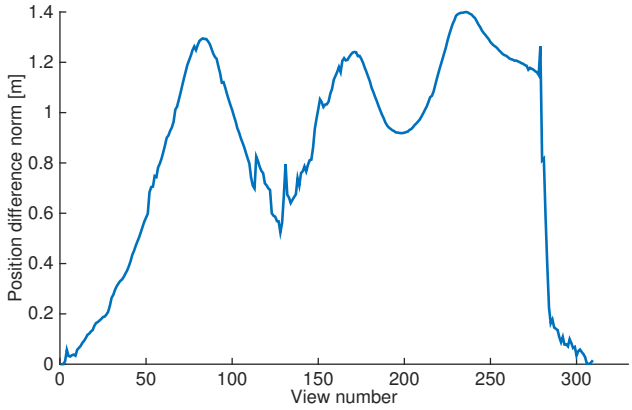
In this section we show computational complexity results of incremental LBA and compare it to other methods using real indoor and outdoor datasets that were collected in our lab. In the first dataset (*Cubicle*) the camera observes a cubicle desk in an open space environment from different viewpoints and distances. In the second dataset, *Outdoor*, the camera follows a trajectory encircling a courtyard and building and performing loop closures as shown in Figure 11. Figure 12 shows typical images from these datasets. Table 1 provides further details regarding the number of views and 3D points, as well as the number of total observations in the two datasets. Figure 1 visualizes the LBA-opitmized camera poses and the reconstructed 3D points based on these camera poses for both datasets.

Image correspondences, as well as the calibration matrices, were obtained by first running Bundler[9] [45] on each dataset. Incremental initializations of

[9] http://phototour.cs.washington.edu/bundler.



(a)



(b)

Figure 11: (a) Estimated trajectory in *Outdoor* dataset. (b) Norm of position difference between LBA and BA. Both figures demonstrate LBA and standard BA produce very similar results.

(a) (b)

Figure 12: Typical images in the *Cubicle* (a) and *Outdoor* (b) datasets.

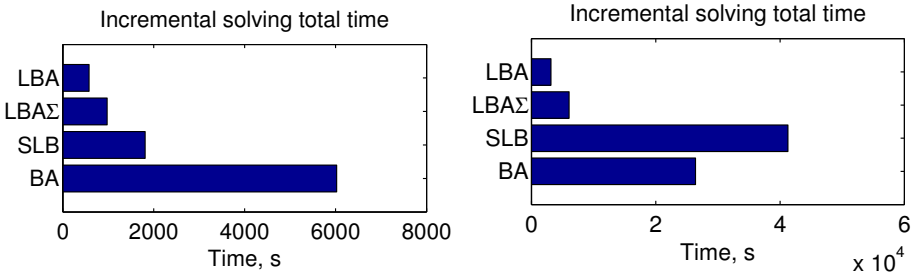
camera poses were calculated by estimating essential matrices between pairs of images, as described in Section 5.2. Degenerate and ill-conditioned camera configurations were identified by having either an under-determined or invalid essential matrix. Initial values for landmarks, required for a standard BA, were computed using triangulation. The bundler solution was not used to initialize any camera poses or landmarks. To eliminate gauge freedoms, the first camera was fixed to the origin and the overall reconstruction scale was determined by constraining the range between the first and second camera to a constant.

We compare the processing time of LBA with BA, as well as with the other structureless BA method: SLB. This method uses the formulation (8) with three-view constraints (10)-(12), which is similar to the cost function used in [47]. Recalling that in LBA, the covariances Σ_i are calculated once and kept fixed (cf. Section 4), we also present a comparison to the case in which these covariances are re-calculated each time a linearization occurs. This variation is denoted by LBAΣ. Incremental smoothing is used in all methods.

The computational complexity in the two real-imagery datasets is shown in Figure 13 and summarized in Table 3. The total time required for all steps of incremental optimization for processing the entire dataset is reported; cameras are added one-by-one to the system for each method, and incrementally optimized using iSAM2 after adding each camera pose.

The results indicate that LBA exhibits much lower computation time than standard BA, approximately by a factor of 10. LBA computation time is similarly lower than SLB (between a factor of 4 and 10). LBA is roughly twice as fast as LBAΣ.

The above observation lines up with the results of the synthetic study (see Section 6), where it was shown that the complexity of LBA increases less dramatically compared to BA when increasing the number of image observations per frame. As seen from Table 1, the average number of observations per frame in the two datasets is on the order of 1000, compared to up to 500 in the synthetic study. These high values of η were obtained by extracting and matching SIFT [33] features by Bundler, as mentioned above.



(a) *Cubicle* (b) *Outdoor*

Figure 13: Comparison of computation time between algorithms for two real-imagery datasets. Times are total for all steps of incremental optimization. Full details are summarized in Table 3.

7 Accuracy of Solution and Uncertainty Estimate

As demonstrated in the previous section, the computational complexity of incremental LBA is significantly reduced compared to other methods. However, this gain in performance comes at the cost of certain compromise in accuracy. In this section we analyze the causes for this compromise in accuracy (Section 7.1), and demonstrate, using the same synthetic and real-imagery datasets from Section 6, that this compromise is minor and high-accuracy results and reliable uncertainty estimates are typically obtained (Sections 7.3-7.5). In particular, we analyze the accuracy of incremental LBA, consider its sensitivity to different aspects and provide a comparison to BA, SLB and LBAΣ (all methods use incremental smoothing).

7.1 Theoretical Probabilistic Analysis

This section analyzes how well the LBA distribution $p_{LBA}\left(X|\mathcal{Z}\right)$ represents the true density $p\left(X|\mathcal{Z}\right)$. An exact calculation of the latter would marginalize the landmarks from the joint $p\left(X,L|\mathcal{Z}\right)$

$$p\left(X|\mathcal{Z}\right)=\int_{L}p\left(X,L|\mathcal{Z}\right)dL. \tag{26}$$

While in practice LBA represents a similar probability density over camera poses as BA, there are two root effects that cause the LBA distribution to be an approximation of the true density: First, *LBA discards some mutual information in large camera cliques*, by considering only the mutual information between camera pairs and triplets introduced by them observing the same landmark. Bundle adjustment, on the other hand, induces mutual information between

all cameras observing the same landmark. Second, *LBA duplicates some information for image measurements used in multiple factors*, double-counting measurements that appear in multiple two- or three-view factors.

As an example of both of these effects, consider observing a landmark l by four views x_1, x_2, x_3 and x_4, as illustrated in Figure 3. The joint pdf is given by

$$p\left(X_4, l|\mathcal{Z}_4\right) \propto \prod_{i=1}^{4} f_{proj}\left(x_i, l\right), \tag{27}$$

where X_4 and $\mathcal{Z}_4$ denote the four camera poses and the four image observations, respectively. On the other hand, the LBA pdf is

$$p_{LBA}\left(X_4|\mathcal{Z}_4\right) \propto f_{2v}\left(x_1, x_2\right) f_{2v}\left(x_2, x_3\right) f_{3v}\left(x_1, x_2, x_3\right) \\ f_{2v}\left(x_3, x_4\right) f_{3v}\left(x_2, x_3, x_4\right) \tag{28}$$

which corresponds to the set of two- and three-view factors (see Figure 3b).

The first effect, discarding of mutual information, can be seen when comparing the LBA pdf with the pdf resulting from eliminating the landmarks from the BA pdf,

$$\begin{aligned} p\left(X_4|\mathcal{Z}_4\right) &= \int_{X_4} p\left(X, L|\mathcal{Z}\right) dX_4 \\ &= p\left(x_1, x_2, x_3, x_4|z_1, z_2, z_3, z_4\right) \end{aligned} \tag{29}$$

The result in the case of BA is a single clique over all camera poses (cf. Figure 3c). In general, there is no way to exactly factor such a dense clique in a way that reduces complexity. The multiple factors of LBA over pairs and triplets (Eq. (28)) reduce complexity instead by discarding some "links" that would otherwise be introduced between cameras.

The second effect, duplication of some image measurement information, can be seen in the sharing of cameras between LBA factors in Eq. (28). Any two factors sharing a camera in common both use the information from the shared camera, effectively duplicating it. For example, $f_{2v}\left(x_1, x_2\right)$ and $f_{2v}\left(x_2, x_3\right)$ both use the information from the measurements in camera 2.

This duplication of information happens since the two- and three-view factors were assumed to have independent noise models, represented by the covariance matrix Σ_i for the ith factor, and therefore could be separately written in the LBA pdf (9). This assumption is violated for factors that share measurements, as in the example above. One approach to avoid double counting is therefore to augment such factors, while accounting for the fact that the same measurement is involved by appropriate cross-covariance terms in the (augmented) covariance matrix. For example, for any two factors representing constraints g_a and g_b, we can define $f_{aug} \doteq \exp\left(-\frac{1}{2}\left\|g_{aug}\right\|^2_{\Sigma_{aug}}\right)$, with $g_{aug} \doteq \begin{bmatrix} g_a & g_b \end{bmatrix}^T$ and the augmented covariance matrix Σ_{aug} :

$$\Sigma_{aug} \doteq \begin{bmatrix} \Sigma_a & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_b \end{bmatrix} \tag{30}$$

where the cross-covariance terms Σ_{ab} are non-zero when the constraints share measurements. However, since multiple factors are combined into a single multi-dimensional factor that involves all the variables in these individual factors, the factor graph becomes denser. Therefore, such an approach is expected to have considerable impact on computational complexity.

As we show in the sequel, despite the above two aspects, the actual LBA distribution is very similar to the true distribution $p(X|\mathcal{Z})$. In future research, we plan to investigate how duplicating information can be avoided while maintaining attractive computational complexity using conservative information fusion techniques. In particular, we envision using covariance intersection [26] to avoid double counting, which would correspond to appropriately modifying the covariance of involved poses while still treating multi-view factors of the same 3D points separately, e.g. without the need to account for Σ_{ab} in Eq. (30). It is worth mentioning that the presented probabilistic analysis is valid for other existing structureless BA methods [47, 41, 18] as well.

7.2 Quantitative Evaluation of Probabilistic Accuracy

While $p(X|\mathcal{Z})$ can be calculated from the BA pdf $p(X, L|\mathcal{Z})$ (see Eq. (26)), this calculation is not tractable in closed form and we therefore use an alternate method to compare the pdfs of LBA and BA. This method evaluates how well LBA and BA agree in both the absolute uncertainty of each camera pose in a global frame, and the relative uncertainty between all pairs of cameras.

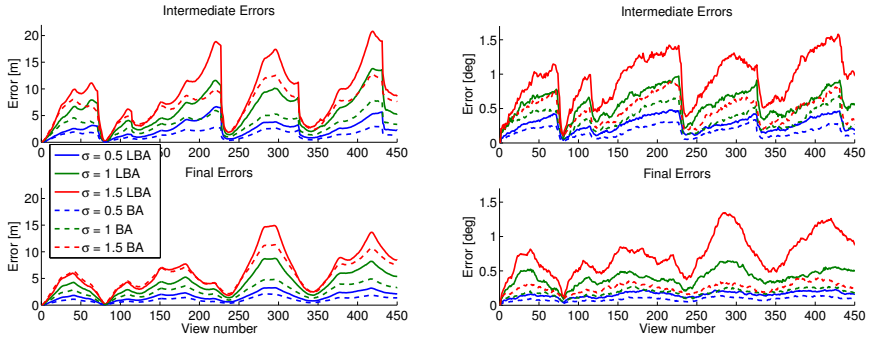
In order to compare uncertainties, we first assume that both $p_{LBA}(X|\mathcal{Z})$ and $p(X|\mathcal{Z})$ are well-approximated as multivariate Gaussian distributions about their MAP estimates

$$p_{LBA}(X|\mathcal{Z}) = N(\mu_{LBA}, \Sigma_{LBA}) \quad , \quad p(X|\mathcal{Z}) = N(\mu, \Sigma)\,. \tag{31}$$

In the following sections, we first discuss accuracy in MAP estimates of LBA (μ_{LBA}), and compare it to MAP estimates of BA (μ), and also to SLB and LBAΣ methods. Results both for synthetic and real-imagery datasets are provided (Sections 7.3 and 7.4, respectively). Using synthetic datasets we also study accuracy sensitivity to different aspects. Finally, in Section 7.5 we examine uncertainty estimation by comparing the estimated covariance matrices in LBA (Σ_{LBA}) and BA (Σ).

7.3 Experiments with Synthetic Datasets

We study the sensitivity of accuracy of both iLBA and incremental BA to the following aspects: different image noise levels, different number of image observations per frame, and dense loop closure observations. These effect of these aspects on computational complexity was already analyzed in Section 6.2.



(a) Position (b) Rotation

Figure 14: Accuracy performance for different levels of image noise ($\sigma \in \{0.5, 1.0, 1.5\}$ pixels). Position (a) and rotation (b) intermediate and final average estimation errors, per camera frame. Shown errors are averaged for 20 runs for each noise level; 200 image observations per frame are used in all cases. While LBA exhibits higher sensitivity to image noise, its processing time is typically smaller, by a factor of 3 in this dataset, compared to BA (see Table 2). Figure best viewed in color.

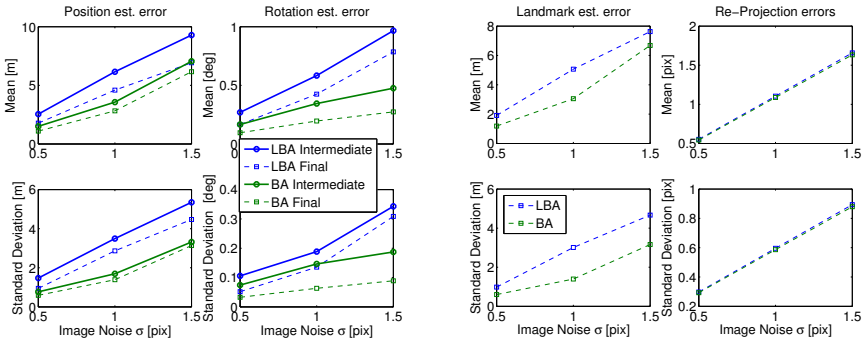
7.3.1 Sensitivity to Image Noise

Position and rotation estimation errors, calculated with respect to ground truth, are shown in Figure 14 for different levels of image noise for the first dataset (Figure 7a). Both intermediate and final estimation errors are shown. Intermediate estimation errors are calculated for each time instant t_k based on all the available measurements $\mathcal{Z}_k$ by that time, while final estimation errors are calculated after the entire dataset has been processed, based on all the measurements of the dataset. The shown values are averages over 20 runs that were performed for each noise level, for each method. 200 image observations per frame are used in all cases.

Figure 15 summarizes the intermediate accuracy results and also presents landmark and re-projection final errors. Table 2 presents the further details of average processing time per frame, total processing time for the entire dataset, and a comparison with SLB and $LBA\Sigma$ methods.

Referring to Figure 14, intermediate estimation errors develop as more images are processed and until loop closure observations are made, which significantly reduce the errors. In contrast, final estimation errors are smoothed, with the maximum error typically located far away from loop closure frames.

For a given image noise level, estimation errors in LBA develop faster than in BA, as long as no loop closures occur. This is expected since the LBA formulation discards some of the available information (cf. Section 7.1). LBA errors are significantly reduced by loop closure observations, usually approaching BA error levels. For image noise $\sigma = 0.5$, LBA intermediate position and



(a) (b)

Figure 15: Accuracy performance for different levels of image noise, calculated for 20 runs of each of the methods. Mean and standard estimation errors are shown, respectively, in the first and second rows. (a) position and rotation intermediate and final estimation errors; (b) Final landmark and re-projection errors.

rotation errors are below 7 meters and 0.5 degrees for all camera poses, and are further reduced in the final solution to values below 3 meters and 0.2 degrees. Normalizing by the total distance traveled (10km), the level of intermediate errors correspond to 0.07% in position and $5e^{-5}$ deg/m in rotation.

Processing time is reported in Table 2 for a typical run with image noise $\sigma = 0.5$. One can observe that LBA is about 3 times faster than BA, and is by far faster than SLB and LBAΣ. The fact that SLB is considerably more computationally expensive is not surprising since it has many more variables to optimize over.

Increasing image noise levels deteriorates accuracy both in BA and LBA: LBA is much more sensitive, since image observations are not corrected, as opposed to SLB method. Therefore, accurate feature correspondences are essential for obtaining good accuracy in LBA. Note that nearly identical re-projection errors are obtained in LBA and BA, for each image noise level (Figure 15b). However, while LBA exhibits higher sensitivity to image noise, its processing time is significantly smaller compared to BA for a given choice of η (number of image observations per frame), see Figures 8 and 13.

Referring to other structureless BA methods, one can observe that SLB yields slightly improved accuracy results, however at a much higher processing time. A very similar accuracy is obtained for LBA and LBAΣ, at a considerably higher processing time of the latter.

Processing time [sec]		BA	SLB	**LBA**	LBAΣ
per frame	μ	1.56	4.07	**0.53**	1.40
	std	1.70	2.81	**0.43**	0.85
Total		779.8	2029	**267.2**	706.5

image noise	Ave. position accuracy [m]			
0.5	1.58	1.71	**2.75**	2.54
1.0	3.57	3.05	**6.16**	6.17
1.5	7.05	5.02	**9.29**	9.29

Table 2: Processing time in an exploration scenario with occasional loop closures (Figure 7a) with image noise $\sigma = 0.5$ pixels and 200 image observations per frame. All methods use incremental smoothing. Processing time is presented per frame (average and standard deviation), and for incrementally processing the entire dataset. Bottom part of the table shows average intermediate position estimation errors for different levels of image noise, comparing between LBA, LBAΣ, BA and SLB methods.

7.3.2 Different Numbers of Image Observations per Frame

The effect of different number of image observations per frame η on accuracy is shown in Figure 16, comparing between LBA and BA. As in the previous section, 20 runs were performed for each η. These runs are summarized in Figure 16 in terms of mean and standard deviation (taken over these runs and over the camera poses in the dataset).

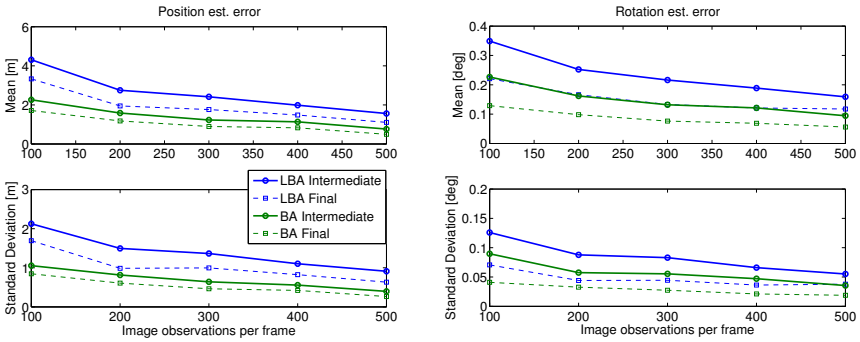
As seen, incorporating more image observations improves the estimation accuracy in both cases, however, as discussed in Section 6 and shown in Figure 8, computational complexity tends to scale more gracefully in LBA and processing time is typically at least 2-3 times faster.

7.3.3 Dense Sequential Loop Closures

Accuracy results for the dense loop closure scenario (Figure 7b) are summarized in Figure 17. Different image noise levels are considered, as in Section 7.3.1. As expected, overall accuracy is improved due to frequent loop closures (compared to results of the previous dataset, Figure 15), with LBA remaining more sensitive to image noise than BA.

7.4 Experiments with Real-Imagery Datasets

In this section we compare the accuracy of LBA, BA, SLB and LBAΣ, all using incremental smoothing, in the two real-imagery datasets (see Figure 7). Specifically, we compare the re-projection errors and the difference in camera pose estimation with respect to BA. In case of structureless BA methods (LBA,



(a) (b)

Figure 16: Effect of different number of image observations per frame on accuracy. Exploration with occasional loop closures scenario is used in all cases. Mean and standard deviation of intermediate and final estimation errors are shown for (a) position and (b) rotation. Accuracy increases as more image observations per frame are used.

LBAΣ, SLB), structure is reconstructed based on the optimized camera poses after convergence.

Figure 18 and Table 3 show the obtained accuracy performance. The cost of the improved timing performance of LBA (analyzed in Section 6.2) is modestly higher errors. Re-projection errors are slightly higher than the other methods. Average position discrepancy of LBA with respect to standard BA is 1.5mm on the *Cubicle* dataset, which covers an area approximately 3m in width. Average position discrepancy on the *Outdoor* dataset is 0.6m, where the area covered is approximately 150m×120m. Rotation discrepancy with respect to standard BA shows the same trend.

In the next section we conclude this analysis by also examining the discrepancy in uncertainty estimates between LBA and BA.

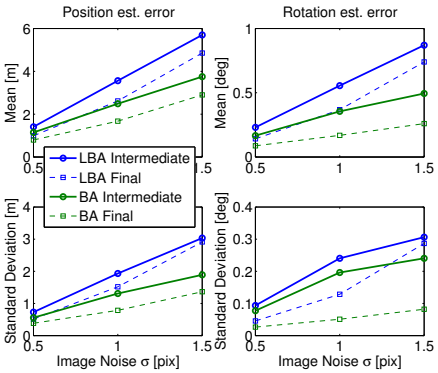
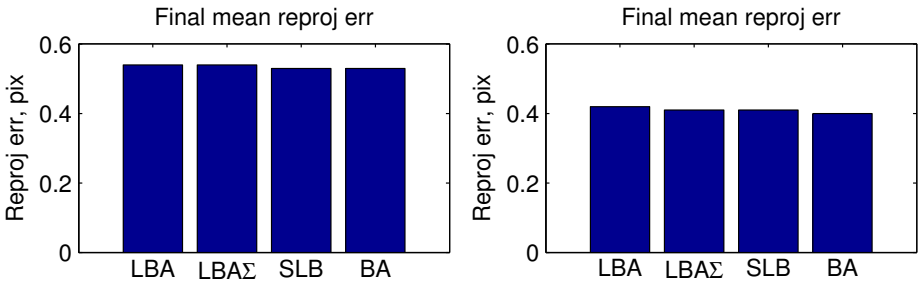


Figure 17: Accuracy performance for different levels of image noise in exploration scenario with dense loop closures. Performance is calculated for 20 runs of each of the methods. Mean and standard deviation of position and rotation intermediate and final estimation errors are shown.



(a) *Cubicle* (b) *Outdoor*

Figure 18: Comparison of reprojection errors between algorithms for two real-imagery datasets. Reprojection errors are computed after incremental optimization of the last camera pose. Full details are summarized in Table 3.

Cubicle	**BA**	**SLB**	**LBAΣ**	**LBA**
Overall time, s	6024	1813	978	**581**
Reproj err, mean	0.53	0.53	0.54	**0.54**
Reproj err, stdd	0.70	0.70	0.71	**0.71**
Avg pos'n diff, m	N/A	.0010	.0015	**.0015**
Avg rot'n diff, rad	N/A	3e-4	4e-4	**5e-4**
χ^2	1.95	1.72	1.95	**1.96**

(a) Cubicle dataset

Outdoor	**BA**	**SLB**	**LBAΣ**	**LBA**
Overall time, s	26414	41284	6076	**3163**
Reproj err, mean	0.40	0.41	0.41	**0.42**
Reproj err, stdd	0.53	0.53	0.53	**0.53**
Avg pos'n diff, m	N/A	0.407	0.376	**0.601**
Avg rot'n diff, rad	N/A	2e-3	1e-3	**1e-3**
χ^2	1.16	0.54	0.78	**1.25**

(b) Outdoor dataset

Table 3: Full result details corresponding to Figure 18. Mean and standard deviation of final reprojection errors, and χ^2 errors. χ^2 error close to 1 indicates that the distribution of the final residuals matches the Gaussian distribution predicted by uncertainty propagation of the image observation noise. Camera pose differences are reported with respect to standard bundle adjustment (BA).

7.5 Experimental Evaluation of Uncertainty Estimate

We study the estimated absolute and relative uncertainties in LBA, both encoded by the estimated covariance matrix Σ_{LBA}, and provide a comparison to BA, using the two real-imagery datasets. We now explain how these estimated uncertainties are compared in practice.

In order to compare relative uncertainty between camera poses, we compare conditional densities $p\left(x_i|x_j,\mathcal{Z}\right)$ between all pairs of cameras. This calculation quantifies how well LBA agrees with BA in relative uncertainty. The conditionals are obtained by integrating out all variables other than x_i and x_j,

$$p\left(x_i|x_j,\mathcal{Z}\right)=\int\limits_{X\setminus\{x_i,x_j\},L} p\left(X,L|\mathcal{Z}\right)/p\left(x_j|\mathcal{Z}\right). \tag{32}$$

This can be done analytically by approximating the joint as a Gaussian around its MAP estimate, and applying sparse factorization,

$$p\left(X,L|\mathcal{Z}\right)=p\left(X\setminus\left\{x_i,x_j\right\},L|x_i,x_j,\mathcal{Z}\right)p\left(x_i|x_j,\mathcal{Z}\right)p\left(x_j|\mathcal{Z}\right), \tag{33}$$

from which the desired conditional $p\left(x_i|x_j,\mathcal{Z}\right)$ can be read off.

Specifically, in our implementation we first calculate the joint marginal information matrix for each camera pair i and j

$$\left[\begin{array}{cc} I_{ii} & I_{ij} \\ I_{ji} & I_{jj} \end{array}\right], \tag{34}$$

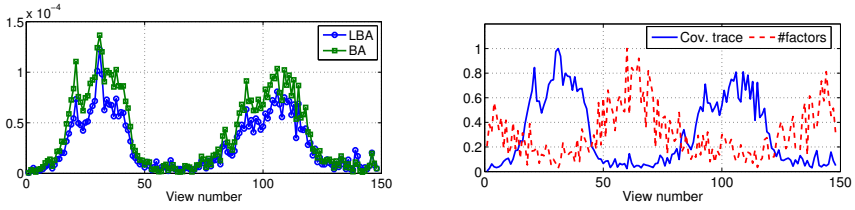
from the appropriate joint pdf ($p\left(X,L|Z\right)$ and $p_{LBA}\left(X|Z\right)$). The information form then allows to conveniently calculate the conditional covariance $\Sigma^{i|j}$ as (see, e.g. [8])

$$\Sigma^{i|j}=I_{ii}^{-1}. \tag{35}$$

We compare the probability density of the camera poses estimated by LBA to that of BA by comparing their discrepancy both in the marginal uncertainty of each camera pose, and in relative uncertainty between each camera pair. To make these comparisons, we define a discrepancy measure of the square roots of the traces of each covariance matrix, which intuitively is an approximate measure of the difference in their standard deviations,

$$\textit{discrepancy}\left(\Sigma_1,\Sigma_2\right)\triangleq c\left(\sqrt{\mathrm{tr}\left(\Sigma_1\right)}-\sqrt{\mathrm{tr}\left(\Sigma_2\right)}\right), \tag{36}$$

where c is a scale factor that converts the unit-less 3D reconstructions into meters. We determined this scalar by physically measuring the dataset collection area, or superimposing the trajectory onto a satellite image. We compute this separately for the blocks of the covariance matrices corresponding to rotation and translation. The units of the discrepancy are radians for rotation ($c=1$) and meters for translation, with c properly determined to correct the reconstruction scale.



(a) (b)

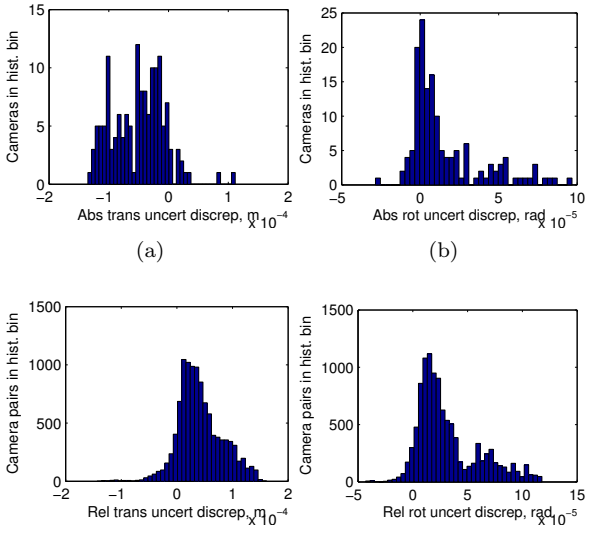
Figure 19: *Cubicle* dataset: (a) Covariance trace of each camera pose. (b) Trace of covariance and number of factors in LBA formulation, both are normalized to 1.

For example, to compare the Gaussian-approximated conditional density of LBA $p_{LBA}\left(x_i|x_j,\mathcal{Z}\right)$ with covariance $\Sigma^{i|j}_{LBA}$ with that of BA $p\left(x_i|x_j,\mathcal{Z}\right)$ with covariance $\Sigma^{i|j}_{BA}$, we compute *discrepancy* $\left(\Sigma^{i|j}_{LBA},\Sigma^{i|j}_{BA}\right)$. Similarly for marginals $p_{LBA}\left(x_i|\mathcal{Z}\right)$ and $p_{BA}\left(x_i|\mathcal{Z}\right)$, we compute *discrepancy* $\left(\Sigma^{i}_{LBA},\Sigma^{i}_{BA}\right)$. A positive discrepancy value means that the uncertainty estimate of LBA is conservative, whereas a negative discrepancy value means that the uncertainty estimate of LBA is overconfident.

A comparison of the absolute uncertainty for the *Cubicle* dataset is given in Figure 19 and Figures 20a-20b. Figure 19a compares, for each camera pose i, between the covariance trace of Σ^{i}_{LBA} and Σ^{i}_{BA}. As seen, the initial uncertainty is very small and it increases as the camera moves around the cubicle desk and drops to low values when the camera captures previously-observed areas thereby providing loop-closure measurements. Figure 19b describes the interaction between the uncertainty of each view and the number of factors that involve this view. As expected, the covariance is higher when less factors are involved and vice versa.

Overall, the absolute uncertainties in LBA and BA are very similar. This can be also observed in Figures 20a-20b that show a histogram of the discrepancy (36) both for position and orientation terms. Typical position discrepancies are near -10^{-4} meters. The discrepancies for relative uncertainties are given in Figures 20c-20d for position and orientation terms.

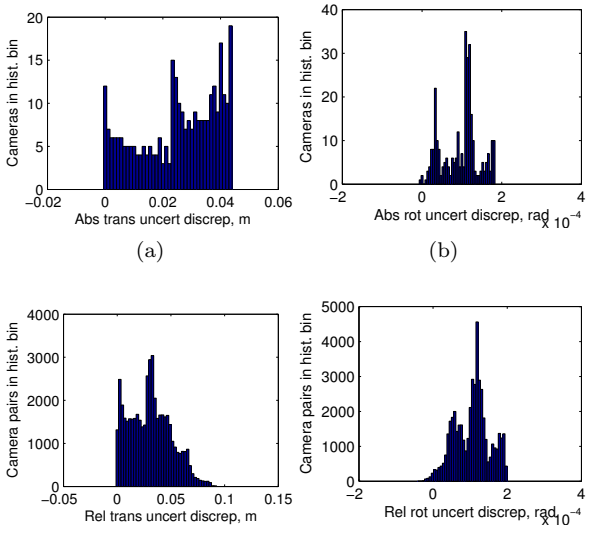
Figure 21 shows the discrepancy histograms for the *Outdoor* dataset. The absolute and relative discrepancies between LBA and BA are small, e.g. less than 5 centimeters in the absolute position for a trajectory that spans an area of 120×150 meters (see Figure 11), and on the order of 10^{-4} radians for the absolute rotation uncertainty.



(a) (b)

(c) (d)

Figure 20: Discrepancy histograms for the *Cubicle* dataset: Absolute position (a) and orientation (b); Relative position (c) and orientation (d) between every camera pair in the sequence.



(a) (b)

(c) (d)

Figure 21: Discrepancy histograms for the *Outdoor* dataset: Absolute position (a) and orientation (b); Relative position (c) and orientation (d) between every camera pair in the sequence.

8 Conclusions

In this paper we presented incremental light bundle adjustment (iLBA), an efficient optimization that is applicable both to structure from motion and robotics applications. The method incorporates two key components to reduce computational complexity. The number of variables in the optimization is significantly reduced by algebraic elimination of 3D points, leading to a cost function that is formulated in terms of three-view constraints and involves only the camera poses. The resulting optimization problem is represented using graphical models and incremental inference is applied to efficiently incorporate new imagery information by exploiting sparsity and using adaptive partial calculations. Typically, only a small subset of past camera poses is recalculated when adding a new camera into the optimization. While 3D points are not explicitly estimated, all or any part of them can be reconstructed based on the optimized camera poses whenever required. Additional sensors, typically available in robotic applications, can be naturally incorporated as well.

A probabilistic analysis of the proposed method was presented, investigating how well the probability distribution of iLBA represents the exact probability over camera poses that is calculated using bundle adjustment. Computational complexity and accuracy of iLBA was studied and compared to incremental bundle adjustment and another recent related method using both synthetic and real-imagery datasets. This study showed that iLBA is considerably faster (by a factor of 2-10) than bundle adjustment and scales more gracefully when increasing the number of image observations per frame. iLBA's accuracy approaches bundle adjustment for reasonable image noise levels. In particular, average difference in position between iLBA and bundle adjustment in an outdoor real-imagery dataset that covers an area of 150m$\times$120m is 0.6 meters. The real-imagery datasets were also used to analyze how well iLBA and incremental bundle adjustment agree in estimated uncertainties, and exhibited very small discrepancies between the two methods.

Adding each new camera pose into the optimization involves instantiating three-view constraints that also involve past camera poses. Which past camera poses are chosen impacts both computational complexity and accuracy of iLBA. Being able to do so optimally and efficiently remains an open problem that will be addressed in future research. Another interesting aspect that is left for future research is comparing computational complexity and accuracy between using multi-view constraints and relative pose constraints. Finally, resorting to GPU-accelerated computing is yet another promising direction to pursue.

Acknowledgments

The authors would like to thank Dr. Stephen Williams, College of Computing, Georgia Institute of Technology, for insightful discussions.

Funding

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.

Appendix

In this appendix we relate between the well-known trifocal constraints and the three-view constraints (10)-(12). Assume some 3D point X is observed by several views. The image projection of this landmark for each of these views is given by $\lambda_i p_i = P_i X$, where λ_i is the unknown scale parameter and $P_i = K_i \begin{bmatrix} R_i & t_i \end{bmatrix}$ is the projection matrix of the i^{th} view.

Choosing the reference frame to be the first view, the projection equations turn into $\lambda_1 K_1^{-1} p_1 = \check{X}$ and $\lambda_i K_i^{-1} p_i = R_i \check{X} + t_i$ for $i > 1$. Here $\check{X}$ denotes inhomogeneous coordinates of the 3D point X. Substituting the former equation into the latter equation eliminates the 3D point, yielding

$$\lambda_i K_i^{-1} p_i = R_i \lambda_1 K_1^{-1} p_1 + t_i \ , \ i > 1 \tag{37}$$

From this point the derivation of three-view and trifocal constraints differs. In the former case, a matrix is constructed from Eq. (37) for the first view and two other views i and j:

$$\begin{bmatrix} R_i K_1^{-1} p_1 & -K_i^{-1} p_i & 0 & t_i \\ R_j K_1^{-1} p_1 & 0 & -K_j^{-1} p_j & t_j \end{bmatrix} \begin{bmatrix} \lambda_1 & \lambda_i & \lambda_j & 1 \end{bmatrix}^T = 0 \tag{38}$$

while in the case of trifocal constraints, the additional scale parameter in Eq. (37) is eliminated by cross multiplying with $K_i^{-1} p_i$. Representing the resulting equations in a matrix formulation yields the so-called multi-view matrix [34]:

$$\begin{bmatrix} \left[K_2^{-1} p_2\right]_\times R_2 K_1^{-1} p_1 & \left[K_2^{-1} p_2\right]_\times t_2 \\ \left[K_3^{-1} p_3\right]_\times R_3 K_1^{-1} p_1 & \left[K_3^{-1} p_3\right]_\times t_3 \\ \vdots & \vdots \end{bmatrix} \begin{pmatrix} \lambda_1 \\ 1 \end{pmatrix} = 0 \tag{39}$$

Enforcing the rank-deficiency condition on the two matrices in Eqs. (38) and (39) yields the three-view and the trifocal constraints [17, 34]. Although Eq. (39) contains expressions for all the views, the resulting constraints relate only between triplets of views.

References

[1] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski. Bundle adjustment in the large. In *European Conf. on Computer Vision (ECCV)*, 2010.

[2] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building Rome in a day. In *Intl. Conf. on Computer Vision (ICCV)*, 2009.

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

[4] Martin Byröd and Kalle Åström. Conjugate gradient bundle adjustment. In *European Conf. on Computer Vision (ECCV)*, pages 114–127, Berlin, Heidelberg, 2010. Springer-Verlag.

[5] 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.

[6] 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.

[7] C. Engels, H. Stewénius, and D. Nistér. Bundle adjustment rules. In *Symposium on Photogrammetric Computer Vision*, pages 266–271, Sep 2006.

[8] R. Eustice, H. Singh, J. Leonard, M. Walter, and R. Ballard. Visually navigating the RMS titanic with SLAM information filters. In *Robotics: Science and Systems (RSS)*, Jun 2005.

[9] 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.

[10] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Commun. ACM*, 24:381–395, 1981.

[11] A. Fitzgibbon and A. Zisserman. Automatic camera recovery for closed or open image sequences. In *European Conf. on Computer Vision (ECCV)*, pages 311–326, 1998.

[12] J.M. Frahm, M. Pollefeys, S. Lazebnik, D. Gallup, B. Clipp, R. Raguram, C. Wu, C. Zach, and T. Johnson. Fast robust large-scale mapping from video and internet photo collections. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(6):538–549, 2010.

[13] V. M. Govindu. Combining two-view constraints for motion estimation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 218–225, 2001.

[14] V.M. Govindu. Lie-algebraic averaging for globally consistent motion estimation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2004.

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

[16] 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.

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

[18] V. Indelman. Bundle adjustment without iterative structure estimation and its application to navigation. In *IEEE/ION Position Location and Navigation System (PLANS) Conference*, April 2012.

[19] 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.

[20] V. Indelman, A. Melim, and F. Dellaert. Incremental light bundle adjustment for robotics navigation. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, November 2013.

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

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

[23] 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.

[24] Y. Jeong, D. Nister, D. Steedly, R. Szeliski, and I.S. Kweon. Pushing the envelope of modern methods for bundle adjustment. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2010.

[25] Y.-D. Jian, D. Balcan, and F. Dellaert. Generalized subgraph preconditioners for large-scale bundle adjustment. In *Intl. Conf. on Computer Vision (ICCV)*, 2011.

[26] S.J. Julier and J.K. Uhlmann. A non-divergent estimation algorithm in the presence of unknown correlations. In *American Control Conference*, pages 2369–73, 1997.

[27] 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.

[28] 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.

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

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

[31] 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.

[32] 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.

[33] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *Intl. J. of Computer Vision*, 60(2):91–110, 2004.

[34] Y. Ma, K. Huang, R. Vidal, J. Košecká, and S. Sastry. Rank conditions on the multiple-view matrix. In *Intl. J. of Computer Vision*, volume 59, pages 115–137, September 2004.

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

[36] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid. RSLAM: A system for large-scale mapping in constant-time using stereo. *Intl. J. of Computer Vision*, 94(2):198–214, 2011.

[37] E Mouragnon, M Lhuillier, M Dhome, F Dekeyser, and P Sayd. Generic and real-time structure from motion using local bundle adjustment. *Image and Vision Computing*, 27(8):1178–1193, 2009.

[38] 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.

[39] K. Ni, D. Steedly, and F. Dellaert. Out-of-core bundle adjustment for large-scale 3D reconstruction. In *Intl. Conf. on Computer Vision (ICCV)*, Rio de Janeiro, October 2007.

[40] D. Nistér. Reconstruction from uncalibrated sequences with a hierarchy of trifocal tensors. In *European Conf. on Computer Vision (ECCV)*, pages 649–663, 2000.

[41] A. L. Rodríguez, P. E. López de Teruel, and A. Ruiz. Reduced epipolar cost for accelerated incremental SfM. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3097–3104, June 2011.

[42] G. Sibley, C. Mei, I. Reid, and P. Newman. Adaptive relative bundle adjustment. In *Robotics: Science and Systems (RSS)*, 2009.

[43] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from Internet photo collections. *Intl. J. of Computer Vision*, 80(2):189–210, November 2008.

[44] N. Snavely, S. M. Seitz, and R. Szeliski. Skeletal graphs for efficient structure from motion. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[45] N. Snavely, S.M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. In *SIGGRAPH*, pages 835–846, 2006.

[46] D. Steedly and I. Essa. Propagation of innovative information in non-linear least-squares structure from motion. In *Intl. Conf. on Computer Vision (ICCV)*, volume 2, pages 223–229, 2001.

[47] R. Steffen, J.-M. Frahm, and W. Förstner. Relative bundle adjustment based on trifocal constraints. In *ECCV Workshop on Reconstruction and Modeling of Large-Scale 3D Virtual Environments*, 2010.

[48] Charles V Stewart. Robust parameter estimation in computer vision. *SIAM review*, 41(3):513–537, 1999.

[49] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, volume 1883 of *LNCS*, pages 298–372. Springer Verlag, 2000.

[50] R. Vidal, Y. Ma, S. Hsu, and S. Sastry. Optimal motion estimation from multiview normalized epipolar constraint. In *Intl. Conf. on Computer Vision (ICCV)*, volume 1, pages 34–41, 2001.

[51] 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.

[52] C. Wu, S. Agarwal, B. Curless, and S.M. Seitz. Multicore bundle adjustment. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3057–3064, 2011.

[53] Z. Zhang and Y. Shan. Incremental motion estimation through local bundle adjustment. Technical Report MSR-TR-01-54, Microsoft Research, May 2001.