Fast Incremental Bundle Adjustment with Covariance Recovery

Viorela Ila
Australian National University, ACRV
Canberra, ACT, Australia
Viorela.Ila@anu.edu.au

Lukas Polok
Apple Inc.
One Infinity Loop, Cupertino, CA, USA
lukas@lukas-polok.cz

Marek Solony
Brno University of Technology
Brno, Czech Republic
isolony@fit.vutbr.cz

Klemen Istenic
University of Girona
Girona, Spain
klemen.istenic@udg.edu

Abstract

Efficient algorithms exist to obtain a sparse 3D representation of the environment. Bundle adjustment (BA) and structure from motion (SFM) are techniques used to estimate both the camera poses and the set of sparse points in the environment. Many applications require such reconstruction to be performed online, while acquiring the data, and produce an updated result every step. Furthermore, using active feedback about the quality of the reconstruction can help selecting the best views to increase the accuracy as well as to maintain a reasonable size of the collected data. This paper provides novel and efficient solutions to solving the associated NLS incrementally, and to compute not only the optimal solution, but also the associated uncertainty. The proposed technique highly increases the efficiency of the incremental BA solver for long camera trajectory applications, and provides extremely fast covariance recovery.

1. Introduction

The field of 3D reconstruction from 2D images is a mature field in computer vision. Available software and applications are able to process and assemble the information from a large amount of images and produce large scale 3D structures; Nevertheless, the majority of the existing applications are designed to be used offline, post-acquisition and do not provide any feedback about the uncertainty of the reconstruction.

Mapping large areas prior to special effect insertion in film production, capturing and documenting underwater reefs, wrecks, thermal vents, etc. are only a few examples in which opportunities to revisit the site for collection of additional images are limited and associated with high increase in operational costs. To diminish the probability of obtaining unsatisfactory captured data from a single session and simultaneously reduce the redundancy of capture efforts, it is important to obtain feedback about the quality of the reconstruction during the acquisition process itself. Such active feedback can be directly utilized by skilled operators or autonomous mission planning schemes to adjust the acquisition mission in progress. As the process is performed online, the methods must be highly efficient.

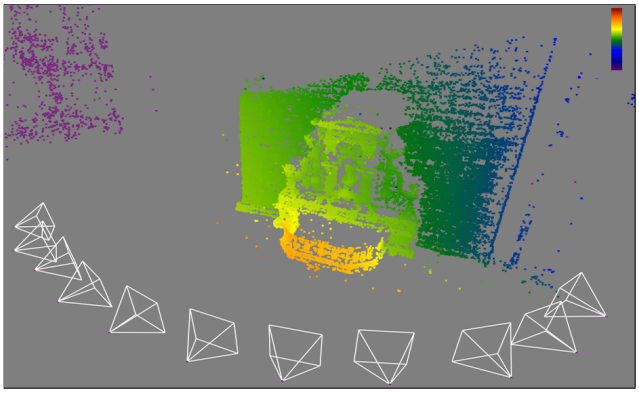

Figure 1: Color-coded marginal covariances of the Fountain-P11 dataset (orange–small uncertainty, violet–large). Despite this dataset is quite small, Google's Ceres solver takes 2556.740 s to recover the covariances. The proposed solution only needs 1.765 s to do that (at each step).

Most of the existing large-scale, image-based 3D reconstruction techniques work by estimating an initial position of cameras and a sparse set of points, followed by a global optimization refinement of the solution. Bundle adjustment

(BA) is the technique which simultaneously estimates the camera poses, calibration parameters and the 3D structure. Large-scale 3D model reconstruction [4] relies on having all the images already available and processing them in batch mode, without providing feedback during acquisition. In general, those problems are regarded as unstructured, the cameras can be placed anywhere in the environment, and the BA step is concerned with the batch optimization of the cameras and 3D points. In contrast, this paper proposes efficient solution to structured 3D reconstruction problems, where the image acquisition is performed sequentially. If the camera is moving, the 3D structure can be obtained by tracking image features over time and subsequently estimating the camera motion [30]. This problem in general relies on sequential BA steps for accurate estimation [10].

In this paper we propose a novel incremental BA technique which naturally adapts to the size of the updates at each step and achieves high performance in processing the images sequentially. This has the advantage that solves, in a principled way, very large scale 3D reconstruction where points move in and, soon after, out of the field of view, without relying on approximations or partial and windowed optimizations. The solution we offer can be plugged to any front-end image processing and automatically restrict the optimization to only the set of variables that need to be optimized, eliminate outlier measures, and provide immediate feedback about the quality of the reconstruction by calculating the marginal covariances. We show that the solution we provide is on average one order of magnitude (almost two in some cases) faster than the high-end solvers.

2. Related Work

Bundle adjustment and structure from motion are similar to simultaneous localization and mapping in robotics (SLAM) [23, 9, 22]. In order to deal with the uncertainty, they are formulated as probabilistic estimation problems and can be solved using nonlinear least squares (NLS) [32].

The paper by Lourakis et al. [25] describes the design and implementation of an efficient NLS solver for BA, with the basic traits shared by most of the other implementations. It makes use of the problem sparsity, and the fact that the BA problem typically contains a relatively large number of independent points. This gives rise to diagonal sub-matrices that make the underlying linear problem easier to solve using Schur complement rather than by applying a general linear solvers directly to the whole matrix as in SLAM [9, 22, 32]. It was shown that using Cholesky factorization on *reduced camera matrix* is efficient for structured, relatively small problems. The paper by Jeong et al. [19] emphasises the importance of block-matrix data-structure and block operations, and variable ordering, ideas also exploited in SLAM by [34]. All those aspects are tackled and efficiently solved by the proposed method. Other techniques to speed up the BA consider solving in a distributed manner [29], although mostly suitable for batch processing. Gradient methods such as preconditioned conjugate gradient (PCG) can be used instead of matrix factorization to solve the linear iterations within NLS [3, 20]. Those methods are mostly used in unstructured problems, where the points are seen from many cameras and the system matrix is far from being block diagonal.

A major challenge appears in online applications, where the state changes every step. One solution to speed up the incremental processing is to reduce the problem to a pose graph optimization where only the poses of key frames are globally optimized and local BA is used to adjust the cameras and the points [28]. Incremental light bundle adjustment [18] is another technique proposed for solving BA incrementally, which it is based on marginalizing out the structure while solving only for the camera poses. Those "light" techniques are not suitable for applications where a feedback about the structure is needed during the acquisition, in here a rather "hard" approach is used to update the structure and its uncertainty every step. Efficient incremental NLS solutions have been developed in the SLAM community, either by working directly on the matrix factorization of the linearized system, by using graphical model-based data structures such as the Bayes tree [22], or by exploiting the sparse block structure of the problems [32]. Those solutions are possible due to small changes in the linearization point at consecutive steps, as well low rank updates directly to the factorized form of the linear system. The matters in bundle adjustment are more complicated, since the increments have much higher rank than in SLAM. Also, when applying the Schur complement trick, the effect of new integrated measurements are hard to be localised, affecting the entire reduced camera matrix.

In a real application, the uncertainty of the estimation plays an important role, in particular in active reconstruction scenarios. This is given by the marginal covariances that encode the uncertainties between a subset of variables. The calculation of the covariance amounts to inverting large matrices, where the resulting matrix is no longer sparse and this can become a computational bottleneck. An exact method for sparse covariance recovery was proposed in [21], based on a recursive formula which calculates any covariance elements on demand from other covariance elements and elements of the Cholesky factorization result. An incremental technique to obtain exact marginal covariances has recently been proposed by Ila et al. [16], and it is based on incremental updates of marginal covariances every time new variables and observations are integrated into the system, and on the fact that, in practice, when the changes in the linearization point are often small and can be ignored. The above-mentioned techniques rely on the Cholesky or a Q-less QR factorizations of the linearized system [5, 13], a

popular approach in the SLAM community. However, the BA and SFM problems have a completely different structure where the number of points is in general much larger than the number of cameras and there are more efficient methods to solve the linearized system.

3. Incremental BA estimation

We can formulate the BA problem as an incremental maximum likelihood estimation (MLE) over a set of camera poses $\mathbf{c} = [c_1 \dots c_{nc}]$ and the points on the structure given by the set $\mathbf{p} = [p_1 \dots p_{np}]$, together forming the state $\boldsymbol{\theta} = [\mathbf{c}, \mathbf{p}]$. We want to find the optimal configuration satisfying the measurements, $\mathbf{z} = [z_1 \dots z_m]$ given by the reprojected points on the image:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ P(\boldsymbol{\theta} \mid \mathbf{z}) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\ -\log(P(\boldsymbol{\theta} \mid \mathbf{z}))\,. \quad (1)$$

Each observation is assumed with zero-mean Gaussian noise with the covariance Σ_k and we measure the reprojection error: $e_k(c_i, p_j, z_k) = z_k - Pr_k(c_i, p_j)$, with $[c_i, p_j] \subseteq \boldsymbol{\theta}$ and $Pr(\cdot)$ is the projection function of a point, p_j, onto the camera c_i, and z_k is the actual pixel measurement. Note that this paper considers only reprojection error minimization, other measurement functions are also possible. Assuming Gaussian noise models, finding the MLE from (1) is done by solving the following NLS problem:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{2} \sum_{k=1}^{m} \left\| z_k - Pr_k(c_i, p_j) \right\|^2_{\Sigma_k}\,. \quad (2)$$

Iterative methods such as Gauss-Newton (GN) or Levenberg-Marquard (LM) are used to find the solution of the NLS in (2). They start with an initial point $\boldsymbol{\theta}^0$ and, at each step, computes a correction $\boldsymbol{\delta}$ towards the solution. For small $\|\boldsymbol{\delta}\|$, a Taylor series expansion leads to linear approximations in the neighborhood of $\boldsymbol{\theta}^0$: $\tilde{\mathbf{e}}(\boldsymbol{\theta}^0 + \boldsymbol{\delta}) \approx \mathbf{e}(\boldsymbol{\theta}^0) + J\boldsymbol{\delta}$, where $\mathbf{e} = [e_1, \dots, e_m]^\top$ stacks all the reprojection errors and J, the Jacobian matrix, which gathers the derivative of the components of $\mathbf{e}$. Thus, at each i^{th} iteration, a linear LS problem is solved

$$\boldsymbol{\delta}^* = \underset{\boldsymbol{\delta}}{\operatorname{argmin}} \frac{1}{2} \left\| A\,\boldsymbol{\delta} - \mathbf{b} \right\|^2\,, \quad (3)$$

where the $A = \boldsymbol{\Sigma}^{-\top/2} J(\boldsymbol{\theta}^i)$ is the system matrix, $\mathbf{b} = \boldsymbol{\Sigma}^{-\top/2}\mathbf{e}(\boldsymbol{\theta}^i)$ the right hand side (r.h.s.) and $\boldsymbol{\delta} = (\boldsymbol{\theta} - \boldsymbol{\theta}^i)$ the correction to be calculated [9]. The minimum is obtained where the first derivative cancels, $A^\top A\,\boldsymbol{\delta} = A^\top \mathbf{b}$, or $\Lambda\boldsymbol{\delta} = \boldsymbol{\eta}$ with $\Lambda = A^\top A$, the square symmetric system matrix approximating the Hessian, and $\boldsymbol{\eta} = A^\top \mathbf{b}$, the right hand side. The solution to this linear system can be obtained either by sparse matrix factorization followed by backsubstitution or by linear iterative methods. After computing $\boldsymbol{\delta}$, the new linearization point becomes $\boldsymbol{\theta}^{i+1} = \boldsymbol{\theta}^i \oplus \boldsymbol{\delta}$. Incremental NLS solving is possible because in real applications, most of the time only a small potion of the increment $\boldsymbol{\delta}$ is relevant, the rest can be ignored [32] and the updates isolated. In BA applications, LM is preferred over the GN methods to deal with local minima, using damping strategies which allow convergence even from poor initial solutions. LM solves a slightly modified normal equation with damping factor λ:

$$(\Lambda + \lambda\bar{D})\boldsymbol{\delta} = \boldsymbol{\eta} \quad \text{or} \quad H\boldsymbol{\delta} = \boldsymbol{\eta}\,, \quad (4)$$

with common choices of $\bar{D} = \operatorname{diag}(\Lambda)$ or $\bar{D} = I$.

Similarly to LM, dog leg (DL) algorithm, which was first described by Powell [35], varies between GN steps and steepest descent (SD) directions [24] with the difference that DL always stays in a *trust region*. The trust region is a hypersphere of radius Δ centred at the current linearization point $\boldsymbol{\theta}^i$ and where the linearized function in (**??**) and nonlinear function behave in a similar way. It is possible to calculate both the GN step $\Lambda\,\boldsymbol{\delta}_{\text{GN}} = \boldsymbol{\eta}$ and also the *direction* of the SD step $\boldsymbol{\delta}_{\text{SD}} \approx \boldsymbol{\eta}$ at the same time. Finally, the dog leg step $\boldsymbol{\delta}_{\text{dl}}$ is chosen as a linear combination of the two, in order to be on or inside the trust radius hypersphere (please, refer to [35, 36] for specific details). Similarly as in LM, the trust radius is modified based on the improvement of the solution once the step has been taken.

The study in [24] shows that DL converges faster than LM while giving solutions of the same quality. In addition to that, DL is appealing from the incremental solving point of view [36], as it does not require modification of the system matrix by damping which would impede incremental factorization updates. In incremental solving, we will attempt to only update the variables which are changing. Altering the entire diagonal of the system matrix would require recalculating much more. DL is also favorable if not only the state mean but also state covariance is needed; then the factorization used by the linear solver can be inverted to get the covariance directly whereas in LM, a second factorization without the damping needs to be formed first.

3.1. Dealing with Outlier Measurements

In 3D reconstruction, a situation often arises when some of the measurements introduced into the system are not affected by normal distributed noise, as assumed in (2) but rather a few of them have a significantly larger error. It is possible to introduce additional variables to the optimized system, which decide on the validity of the measurements [42]. Alternatively, it is possible to calculate the weights directly, without any additional variables as in [1] by using standard *robust estimators*.

The appealing property of robust estimators or M-estimators (maximum likelihood type estimators) [15] is their simple integration into the ordinary nonlinear least

squares framework. In fact, NLS is a special case of an M-estimator where the loss function $\rho(\mathbf{b})$ happens to be the L2 norm or squared Mahalanobis norm. Other loss functions that are less susceptible to the outliers are possible [15], we used Huber's pseudo-L1 function. To integrate it in the NLS framework, each observation z_k is assigned a weight given by $w_k = \frac{\psi(\mathbf{b}_k)}{\mathbf{b}_k}$, with $\psi(\mathbf{b}) = \frac{d\rho(\mathbf{b})}{d\mathbf{b}}$. These weights then multiply the measurement covariances Σ_k in (2).

In the literature, estimating the scale of the problem is often discussed, since the scale s of the errors $\tilde{\mathbf{b}} = \frac{\mathbf{b}}{s}$ follows the general scale of the problem. There are methods to estimate this scale, such as median absolute deviation (MAD) or Huber's "alternative proposal" [15]. In the context of incremental solving, adjusting the scale estimate leads to the change of the weights for all the observations and thus impedes incremental updates. The scale could either be estimated with hysteresis or guessed apriori from the sensor characteristics and fixed, an approach we also adopted.

3.2. Variable Parametrization

There are several approaches to parameterizing the BA problem. Landmarks can be parametrized in *Euclidean coordinates*, $\mathbf{p}_i = [x_j, y_j, z_j]^\top$ resulting in a 3D vector. Scale can be introduced as a parameter resulting in a 4D vector of homogeneous coordinates $\mathbf{p}_j = q_j[x_j, y_j, z_j, 1]^\top$, where q is the scale factor. A commonly used factor is $q_j = 1/z_j$, which yields the *inverse depth* parameterization of the points, resulting in a 3D vector $p_j = [x_j/z_j, y_j/z_j, 1/z_j]^\top$.

The camera poses can be parameterized using 6D vectors. It is common to consider a camera pose as an element of the Lie algebra $\widehat{c}_i \in \mathfrak{se}(3)$ of the special Euclidean group $SE(3)$ with $\widehat{c}_i$ being the matrix form of the pose $c_i = [v, \omega]^\top$, $c_i \in \mathbb{R}^6$, with $\omega \in \mathbb{R}^3$, the rotation component and $v \in \mathbb{R}^3$ the translation component. The scale can be better estimated during the optimization process by considering the camera poses as elements of the Lie algebra $\widehat{c}_i \in \mathfrak{sim}(3)$ of the Similarity group $Sim(3)$, $c_i = [v, \omega, q]^\top$, $c_i \in \mathbb{R}^7$, $q \in \mathbb{R}$ being the scale factor [39].

Every point $\mathbf{p}_j$ from the state vector $\boldsymbol{\theta}$ can be expressed either globally in the world coordinate frame or locally in coordinate frame of one of the cameras. In the same way the cameras can be represented relative to previous camera or in global coordinates. Polok et al. [33] provides an exhaustive analysis of the effects of different points and camera parameterizations when solving the problem incrementally. In incremental updates, the number of variables affected by the update has a direct impact on the performance. They concluded saying that using local inverse depth parameterization the number of variables affected by the updates stays bounded, even if the size of the state cotinuously grows.

3.3. Solving using Schur Complement

A permutation can isolate the cameras C from the points P, yielding this system matrix and r.h.s.:

$$\Lambda = \begin{bmatrix} C & B \\ B^\top & P \end{bmatrix} \; ; \quad \boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{\eta}_C \\ \boldsymbol{\eta}_P \end{bmatrix} \tag{5}$$

It is common to use Schur complement to invert such matrix:

$$S_{\Lambda/P} = S_C = C - BP^{-1}B^\top \tag{6}$$

$$S_{\Lambda/C} = S_P = P - B^\top C^{-1} B \, . \tag{7}$$

We call $S_{\Lambda/P}$ in (6) the Schur complement (SC) corresponding to the camera poses, a sparse matrix with the size given by the number of cameras, and $S_{\Lambda/C}$ in (7) the SC corresponding to the points. SC is in general not as sparse as Λ so calculating $S_{\Lambda/C}$ is often intractable, due to the fact that the number of points is much larger than the number of cameras. Only $S_{\Lambda/P}$ will be considered in our derivations, and the remainder of this paper, it will be denoted simply with S_C. In order to solve the normal equation we apply:

$$(C - BP^{-1}B^\top)\, \boldsymbol{\delta}_C = \boldsymbol{\eta}_C - (BP^{-1})\, \boldsymbol{\eta}_P \tag{8}$$

$$\boldsymbol{\delta}_P = P^{-1}(\boldsymbol{\eta}_P - B^\top \, \boldsymbol{\delta}_C) \tag{9}$$

3.4. Incremental Schur Complement

This paper shows that in many incremental BA applications, it is useful to reuse, if possible, the computations performed in the previous steps. In case small changes in the linearization point can be neglected, additive incremental updates on the Schur complement can be performed:

$$\widehat{\Lambda} = \begin{bmatrix} C & B \\ B^\top & P \end{bmatrix} + \begin{bmatrix} \triangle C & \triangle B \\ \triangle B^\top & \triangle P \end{bmatrix} = \begin{bmatrix} \widehat{C} & \widehat{B} \\ \widehat{B}^\top & \widehat{P} \end{bmatrix} \, . \tag{10}$$

The number of points seen by the new camera and which constitute the update, is in general much smaller than the total number of points, therefore the matrices $\triangle C, \triangle B$ and $\triangle P$ have low rank and most of their elements are zero. Assuming the points are independent, P and $\widehat{P}$ are block-diagonal matrices, and therefore easy to invert. Here, it is important to note that LM nonlinear optimizers cannot be used, the damping parameter λ would induce changes in the entire diagonal of C and P. Therefore, a dog leg optimizer is assumed.

Taking the difference $\triangle(P^{-1}) = \widehat{P}^{-1} - P^{-1}$ after the inverse, the updated Schur complement of (10) becomes:

$$\begin{aligned} \widehat{S}_C &= \widehat{C} - \widehat{B}(P^{-1} + \triangle(P^{-1}))\widehat{B}^\top \\ &= \widehat{C} - BP^{-1}B^\top - BP^{-1}\triangle B^\top - \\ &\quad \triangle B P^{-1}\widehat{B}^\top - \widehat{B}\triangle(P^{-1})\widehat{B}^\top \\ &= S_C + \triangle C - BP^{-1}\triangle B^\top - \\ &\quad \triangle B P^{-1}\widehat{B}^\top - \widehat{B}\triangle(P^{-1})\widehat{B}^\top \, . \end{aligned} \tag{11}$$

Note that $\triangle(P^{-1})$ is nonzero only in the new or changing blocks, while $\triangle(P^{-1}) \neq (\triangle P)^{-1}$. Taking advantage of symmetry $P^{-1} = P^{-\top}$ we can write:

$$\begin{aligned}\triangle S_C &= \triangle C - (\triangle B P^{-1} \widehat{B}^\top - \triangle B P^{-1} \triangle B^\top)^\top - \\ &\quad \triangle B P^{-1} \widehat{B}^\top - \widehat{B} \triangle (P^{-1}) \widehat{B}^\top \\ &= \triangle C - \widehat{B} F + \triangle B F - F^\top \widehat{B}^\top - \widehat{B} \triangle (P^{-1}) \widehat{B}^\top \\ &= \triangle C - (\widehat{B} - \triangle B) F - \left(F^\top + \widehat{B} \triangle (P^{-1}) \right) \widehat{B}^\top, \quad (12)\end{aligned}$$

where $\widehat{S}_C = S_C + \triangle S_C$ and the common subexpression $F \triangleq P^{-1} \triangle B^\top$ is sparse, with the same sparsity pattern as $\triangle B^\top$ and is explicitly evaluated only once. That way, the number of terms with products is reduced from three in (11) to two. The additions in the left sides of the two product terms in (12) also help to save some operations as the summands have similar sparsity patterns and so the number of nonzeros in the sum does not exceed the number of nonzeros of the denser of the summands.

The cost of calculating the Schur complement from scratch is dominated by the cost of the product $\widehat{B} \widehat{P}^{-1} \widehat{B}^\top$ that is not easily described in terms of the sizes of the matrices and their numbers of nonzeros, as it depends heavily on the sparsity pattern of $\widehat{B}$, given by the visibility of the same landmarks by different cameras. Therefore, many pairs of columns have nonzero dot product, which in turn yields dense SC (e.g. for the Venice dataset, S_C is over 40% dense).

The problematic term in (12) is $\widehat{B} \triangle (P^{-1}) \widehat{B}^\top$ that has the same complexity as calculating the Schur complement from scratch if $\triangle(P^{-1})$ has the same rank as $\widehat{P}$ (and this happens if all the variables are changing). Therefore, it is necessary to switch between the proposed incremental updates and sometimes resort to a batch step. A benchmark of the costs was made on all the datasets described in 4, and can be seen in 2. The horizontal axis is the rank of the variables being updated and the vertical axis is the speedup over a batch step. It can be seen that the cost is smaller if fewer than 50% of the variables are being updated.

In order to solve the system in (8) the right hand side (r.h.s.) of the equation must be evaluated. This involves the product of large matrices with a vector $(B P^{-1})\, \boldsymbol{\eta}_P$. For the updated system this becomes:

$$\begin{aligned}\widehat{\mathbf{r}} &= \widehat{B} \widehat{P}^{-1}\, \widehat{\boldsymbol{\eta}}_P \quad (13)\\ &= (B + \triangle B)(P^{-1} + \triangle(P^{-1}))\boldsymbol{\eta}_P + \widehat{B} \widehat{P}^{-1} \triangle \boldsymbol{\eta}_P \\ &= B P^{-1} \boldsymbol{\eta}_P + \widehat{B} \triangle (P^{-1}) \boldsymbol{\eta}_P + \triangle B P^{-1} \boldsymbol{\eta}_P + \\ &\quad \widehat{B} P^{-1} \triangle \boldsymbol{\eta}_P + \widehat{B} \triangle (P^{-1}) \triangle \boldsymbol{\eta}_P \\ &= \mathbf{r} + \widehat{B} \triangle (P^{-1}) \boldsymbol{\eta}_P + \triangle B P^{-1} \boldsymbol{\eta}_P + \widehat{B} \widehat{P}^{-1} \triangle \boldsymbol{\eta}_P \,.\end{aligned}$$

Observe that by bookkeeping $\boldsymbol{\eta}_P$ and $P^{-1} \boldsymbol{\eta}_P$ vectors, the rest of the computations are just low rank multiplications. The cost of computing the r.h.s. from scratch is $O(2\text{nnz}_{\widehat{P}^{-1}}) + O(2\text{nnz}_{\widehat{B}})$ while incrementing it costs $O(6\text{nnz}_{\triangle B}) + O(4\text{nnz}_{(\triangle P)^{-1}})$. Therefore, incrementing is cheaper if $\text{nnz}_{\triangle B} < \text{nnz}_{\widehat{B}}/3$.

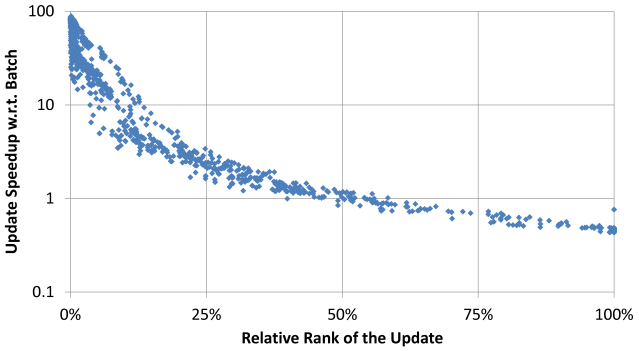

Figure 2: Incremental Schur complement speedup.

Once both the SC and the r.h.s. are updated, the system can be solved using an off-the-shelf linear solver. It could be possible to apply incremental Cholesky factorization to the Schur complement. However, unlike in incremental SLAM where the changes in the system matrix are localized, the changes to the Schur complement are not. For that reason, an ordinary batch sparse block Cholesky factorization is used instead of its incremental variants [22, 32].

3.5. Covariance Calculation

The uncertainty of the variables is encoded in the covariance matrix $\Sigma = \Lambda^{-1}$. For the ordering applied to Λ in (5), we have a corresponding partition for Σ:

$$\begin{bmatrix} C & B \\ B^\top & P \end{bmatrix} \cdot \begin{bmatrix} \Sigma_C & \Sigma_B \\ \Sigma_B^\top & \Sigma_P \end{bmatrix} = \begin{bmatrix} I_c & 0 \\ 0^\top & I_p \end{bmatrix}, \quad (14)$$

where I_c and I_p are the identity matrices of the size $nc \times nc$ and $np \times np$, respectively. The covariances of the camera poses and the points, respectively, can be calculated naively using the Schur complement:

$$\Sigma_C = (C - B P^{-1} B^\top)^{-1} = S_C^{-1} \quad (15)$$

$$\Sigma_P = (P - B^\top C^{-1} B)^{-1} = S_P^{-1} \,. \quad (16)$$

Applying the Woodbury formula for Σ_P we obtain:

$$\begin{aligned}\Sigma_P &= P^{-1} + P^{-1} B^\top (C - B P^{-1} B^\top)^{-1} B P^{-1} \\ &= P^{-1} + P^{-1} B^\top\, S_C^{-1}\, B P^{-1} \\ &= P^{-1} + P^{-1} B^\top \Sigma_C B P^{-1} \,. \quad (17)\end{aligned}$$

The computation and memory use can be reduced by factorizing the Schur complement of the camera matrix:

$$\Sigma_C = S_C^{-1} = T^{-1}\, T^{-\top}, \quad (18)$$

where T^{-1} is a sparse triangular matrix with the storage requirements similar to that of $\mathrm{chol}(S_C)$. Note that a different factorization may be used here, e.g. $LDL^\top$ or LU, in order to improve numerical stability, and in exchange for little extra space and time. With that, (17) transforms to:

$$\begin{aligned}\Sigma &= P^{-1} + P^{-1}B^\top\, T^{-1}\, T^{-\top}\, BP^{-1} \\ &= P^{-1} + V^\top V\,, \end{aligned} \tag{19}$$

where $V = T^{-\top}\, BP^{-1}$ is only used here to illustrate symmetry and is not explicitly formed. One block element of the diagonal of the covariance matrix can be computed as:

$$\Sigma_{P_{ii}} = P_{ii}^{-1} + V_{i,:}^\top\, V_{:,i}\,. \tag{20}$$

Computation can be saved by taking advantage of sparsity of the matrices and the fact that most applications require only the recovery of the block diagonal of Σ_P.

4. Experimental Results

This section evaluates the proposed incremental BA framework. For the implementation of the proposed algorithm we selected an existing implementation of a NLS solver. Several such choices exist in the SLAM community [23, 22, 32]. We selected SLAM++ implementation based on the exhaustive evaluations in [32, 16] and the fact that it facilitates the manipulation of the matrix incremental updates. For comparison two popular NLS solvers in computer vision, g2o and Ceres, were used.

g2o: General Framework for Graph Optimization is a popular framework for nonlinear optimization in robotic vision [23]. It contains several optimizers, based on GN, LM or (experimental) DL. While designed to be easily extensible, g2o can solve BA and SLAM problems out-of-the-box. The BA implementation is restricted to batch solving, yet an incremental SLAM solver is available. The support for robust solving is also implemented. It can also recover covariances, using the same recursive formula implementation as described in [5, 13, 21]. However, it needed some code modifications to be able to recover covariances of the landmarks, as it can only recover the covariances of the poses. Additionally, it cannot add new variables to a Schur-complemented system so it was necessary to rebuild the system from scratch with every new camera added. This was not included in the timing for the sake of fairness.

Ceres: Google's solver [2] received much attention, as it is used in their 3D Maps and Street View applications. It is mostly focused on batch solving, using a variety of available algorithms (GN, LM, DL, subspace DL [6], CG). It relies on SuiteSparse [8] and Eigen [14] for solving the linear systems via a set of sparse and dense solvers. It supports automatic and numeric derivatives, as well as analytical ones. It also has a multitude of robust loss functions. Ceres can also recover covariances of the solution, either using dense SVD or using sparse QR decomposition of the system matrix Λ followed by sparse right-hand-side backsubstitution.

SLAM++: implements incremental solutions for SLAM [32, 17] using sparse block Cholesky factorization and recently also SFM [33] and has the advantage of highly efficient block matrix data-structure that facilitates structural and numerical changes of block matrices as well as arithmetic operations. Both the CPU and the GPU versions were shown to be faster than the SuiteSparse variants of element-wise implementations [31]. This library sparked our attention because of its two order of magnitude faster SLAM covariance recovery compared to any existing solvers [16], therefore we identified it as being the best candidate for the base implementation of our proposed algorithm for incremental Schur complement solver with covariance recovery. Note that the computations in incremental BA highly differ from SLAM as shown is section 3.3 and 3.5. The incremental block Cholesky factorization implemented in SLAM++ [32, 17] is not suitable for solving BA problems, it does not support LM nor DL solving and fails once the system matrix becomes positive indefinite. Neither it supports solving using Schur complement which is known to be much faster. Finally, it relies on constrained ordering of the variables (last pose and corresponding landmarks are ordered last), the complexity directly depends on the rank of the update. This rank is prohibitively high with BA.

All the benchmarks were ran on a computer with Intel Core i7-4790 which is an eight-core CPU running at 3.6 GHz, equipped with 16 GB of RAM. It had Ubuntu 14.04 and g++ 4.8. At the time of running the benchmarks, there were no other applications running in the background. Each test was run ten times and the results were averaged. Some of the benchmarks of covariance recovery of [16] required more than 16 GB of memory and were run on a computer with Intel Xeon E5-4627v2 at 3.3 GHz and 256 GB of RAM. Results on this machine are clearly marked with a dagger†.

The compared implementations were evaluated on one simulated and five standard real datasets. The simulated dataset *Loop* consists of 180 camera poses and 3600 3D points. The real datasets include *Fountain-P11* [40], *New College* [37], *TUM Frei3* [41], *Guildford Cathedral*[1] and the *Fast & Furious 6* dataset which was kindly provided by Double Negative Visual Effects[2].

The images were processed using the publicly available OpenMVG library [27] which was modified to handle incremental processing. As the structure (3D points) and motion parameters are inferred entirely from the projections of the points on the images, salient features are robustly detected and tracked from previous images. The association of fea-

[1] can be obtained from `http://cvssp.org/impart/`

[2] `http://www.dneg.com/`

tures uses Lowe's ratio test and geometrical filtering (i.e. epipolar constraints) inside the parameter-free robust statistical method Acontrario-RANSAC [26] to prevent the influence of possible outliers on the estimation of the geometrical model. Using the estimated camera pose, the new 3D points can be triangulated by exploiting features matched with candidates from previous images. In this way, the new camera together with the new points are intialized and can be added into the system.

In order to guarantee repeatability and fairness of the evaluation, the same input data is used by all the solvers. When processing the images as decsribed above, the data is saved into a *graph file* where the estimated camera poses and the 3D points are referred to as *vertices* and the corresponding observations as *edges*. All the solvers use the same graph files as input (save for minor conversions required by the solvers, i.e. changing YPR to quaternions). The time spent in creating these graph files is not counted, as it is the same for all the solvers.

Incremental Schur complement performance: For linear solving we compared the batch Schur complement solvers in SLAM++, g2o and Ceres to the proposed incremental Schur complement that was implemented on top of SLAM++. For g2o and SLAM++, LM solvers with ten iterations per every new camera were used, since neither of these solvers have a mature dog leg implementation. The proposed method is based on DL. For Ceres, the DL optimizer with equivalent settings was used. The incremental solvers in SLAM++ are intended for SLAM graph optimization [32, 17] and are *not suitable* for solving BA, so they were not considered for this comparison.

The cost of the proposed method depends on the sparsity of the update. When performing only a single iteration at each step, the updates are typically very dense and the proposed method has the same results as batch. Therefore, we let the nonlinear solver iterate until convergence at each step, to get the asymptotic time. The other implementations perform batch solving where all the iterations are equally expensive and so it is possible to multiply the time by the desired number of iterations. The landmarks are represented in relative coordinates and every camera pose is $\widehat{c}_i \in \mathfrak{sim}(3)$. Robust outlier rejection strategy described in 3.1 is used. All the tests use Huber loss function of width 16 px. The results are in 1 and show that the incremental Schur complement methods perform better in applications with long camera trajectories (*Loop*, *New College*, *TUM Frei3* and *Fast & Furios 6*); in *Fountain-P11* and *Cathedral* the density is higher therefore the incremental solver is not advantageous. In *Fountain-P11* specifically, the solver takes several bad steps while trying to fully converge, which slows it down. We plan to address this issue in the future. Note that SLAM++ also supports GPU acceleration but we only had GeForce GTX 970 which has very low

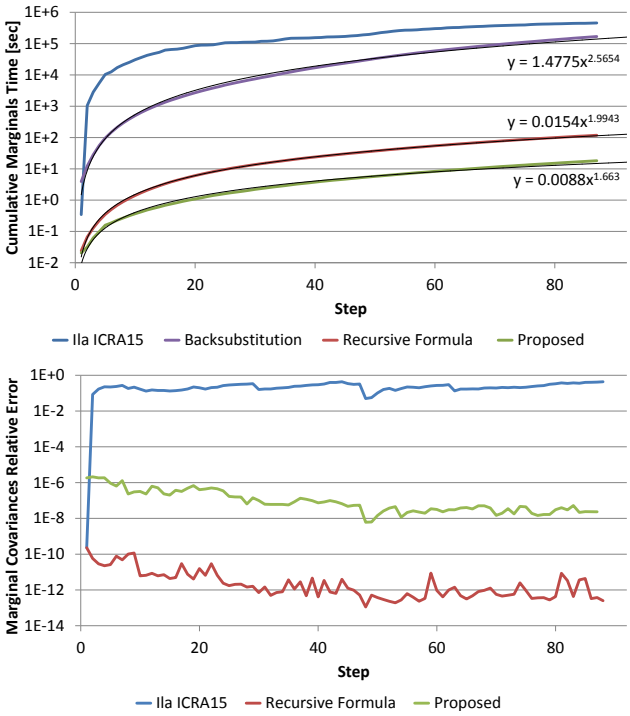

Figure 3: Incremental covariance recovery analysis on *Cathedral*; top–time evaluation, bottom–precision evaluation.

performance in double-precision numbers and we obtained only a modest speedup. The times measured with this GPU were omitted to save space.

Covariance recovery performance: The proposed method for recovering block diagonal of the covariance matrix was tested on several datasets and reported in 2. The full covariance is a dense matrix and for large systems, it is prohibitive to recover all its elements. The marginal covariances on the block diagonal are required by most of the applications. Elements of the covariance matrix are calculated differently by each solver. For Ceres, the SuiteSparse QR option was used, being the most adequate for sparse problems. For g2o, recursive formula was used to obtain only the block diagonal elements. Note that, in both cases, an extra Cholesky factorization of Λ is required. SLAM++ has a more efficient implementation of the recursive formula, accompanied by the incremental updates [16]. While suitable for SLAM problems where the rank of the update is typically very small, for BA it requires a dense inverse of the innovation matrix that requires large amounts of space *and time*, as reported in the table. The proposed covariance calculation reusing the Schur complement in (20) does not require any extra factorization and has low asymptotic cost. The recovered covariances were compared to reference calculated using backsubstitution by taking L2 norm of the difference.

A comparative evaluation of the cost and precision of the recursive formula implementation in SLAM++, method

Dataset	Loop	Fountain-P11	New College	TUM Frei3	Cathedral	F&F 6	Boreas*
Num. of Cameras/Points	180/3600	11/58817	224/26935	326/40435	92/57957	160/136453	1772/170018
Num. of Observations	119642	171026	87532	134620	422163	466262	455776
Schur-g2o	478.073 s	**21.700** s	335.656 s	635.853 s	685.928 s	1282.380 s	19578.700 s
Schur-Ceres	317.080 s	36.258 s	92.106 s	1047.801 s	**483.320** s	955.074 s	15138.994 s
Schur-SLAM++	265.511 s	149.331 s	50.418 s	345.821 s	739.483 s	1158.909 s	20876.609 s
incSchur-Proposed (12)	**191.454** s	69.882 s	**13.668** s	**119.671** s	705.738 s	**889.026** s	**9541.680** s
Num. of Iterations*	755/179	112/19	305/223	1275/325	910/91	1590/159	35490/3549
Schur-g2o (RMSE)	0.796 px	0.684 px	2.639 px	8.789 px	3.208 px	1.381 px	2.616 px
Schur-Ceres (RMSE)	1.787 px	0.303 px	5.730 px	7.722 px	14.028 px	7.722 px	8.466 px
Schur-SLAM++ (RMSE)	0.795 px	0.620 px	3.223 px	9.108 px	2.639 px	1.307 px	12.575 px
incSchur-Proposed (RMSE)	0.795 px	0.616 px	2.884 px	8.479 px	2.644 px	1.557 px	6.022 px

Table 1: Nonlinear solving performance (best times in bold). *The number of iterations for the proposed incremental until convergence ($\boldsymbol{\delta} < 0.005$) / number of steps. *Boreas is an asynchronous multi-cameras rig dataset.

Dataset	Loop	Fountain-P11	New College	TUM Frei3	Cathedral	F&F 6
Time proposed (20)	**13.590** s	**1.765** s	**20.050** s	**62.240** s	**18.146** s	**61.827** s
Error w.r.t. backsubstitution	$1.91 \cdot 10^{-8}$	$1.83 \cdot 10^{-4}$	$3.58 \cdot 10^{-9}$	$3.51 \cdot 10^{-9}$	$5.35 \cdot 10^{-7}$	$2.83 \cdot 10^{-7}$
Memory proposed	5.23 MB	1.78 MB	2.91 MB	4.88 MB	4.73 MB	6.40 MB
Ceres backsubstitution	3014.174 s	2556.740 s	6370.129 s	28926.793 s	34926.434 s	170835.508 s
g2o recursive formula	1029.138 s	10.052 s	233.087 s	456.748 s	808.790 s	999.471 s
Time [16]	2422.103 s	73.435† h	988.436 s	4824.560 s	126.919† h	80.979† h
Memory required in [16]	207.16 MB	23.66 GB	686.29 MB	1.80 GB	18.64 GB	24.96 GB
Time SLAM++ recursive formula	62.429 s	17.921 s	227.909 s	65.851 s	117.904 s	165.532 s

Table 2: Timing results of the evaluated covariance recovery methods (best times in bold). †Some of the runs required too much memory and had to be calculated on a slightly different system with 256 GB of RAM, please refer to section 4 for more details.

of [16] and the proposed method is shown in 3. Note that the proposed method is significantly faster while also being precise and that the final value of the recursive formula (117.904 s) is still several times lower than that of g2o or Ceres in 2. The method of [16] is imprecise due to handling extremely large dense matrices and calculating dot products of extremely long vectors. Special treatment would be needed to make it numerically stable in such conditions.

5. Conclusions

This paper proposed methods to efficiently obtain incremental updates of the BA solution every step a camera and corresponding points are incorporated into the system. We propose an incremental Schur complement formulation that brings up to threefold reduction in solving time if the updates are sparse. If the updates are dense, it gracefully degenerates to batch solving. Moreover, a highly efficient covariance recovery technique was integrated, and was shown to be more than one order of magnitude faster than the existing implementations while having only modest memory requirements. This is the first existing solver that can provide the covariances of the estimate in BA at the cost comparable to that of NLS solving. Both features are extremely important in applications where an immediate feedback about the 3D reconstruction is required to guide the acquisition. In the future, we plan to perform more evaluations on a Tesla-class GPU and to see if the proposed method could be adapted to commodity hardware and perhaps even for embedded GPUs onboard robotic platforms. Most of the time in calculating the marginals is spent in (20) which is a simple sparse matrix multiplication and a reasonable speedup can be expected from a GPU implementation. The implementation of the proposed method can be found at `http://sf.net/p/slam-plus-plus/`.

6. Acknowlegments

This research was supported by the ARC through the "Australian Centre of Excellence for Robotic Vision" CE140100016 and by the Ministry of Education, Youth and Sports of the Czech Republic from the NPU II project IT4Innovations excellence in science (LQ1602), the TA-CR Competence Centres project V3C Visual Computing Competence Center (no. TE01020415) and research project no. VI20172020068.

References

[1] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard. Robust map optimization using dynamic covariance scaling. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 62–69. IEEE, 2013.

[2] S. Agarwal and K. Mierle. Ceres solver. `http://ceres-solver.org/`, 2012.

[3] S. Agarwal, N. Snavely, S. Seitz, and R. Szeliski. Bundle adjustment in the large. In *Computer Vision – ECCV 2010*, pages 29–42. 2010.

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

[5] A. Björck. *Numerical methods for least squares problems*. SIAM, 1996.

[6] R. H. Byrd, R. B. Schnabel, and G. A. Shultz. Approximate solution of the trust region problem by minimization over two-dimensional subspaces. *Mathematical Programming*, 40(1–3):247–263, 1988.

[7] M. Byröd and K. Åström. Conjugate gradient bundle adjustment. In *Eur. Conf. on Computer Vision (ECCV)*, pages 114–127. Springer Heidelberg, 2010.

[8] T. Davis. Csparse. `http://www.cise.ufl.edu/research/sparse/SuiteSparse/`.

[9] 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.

[10] C. Engels, H. Stewénius, and D. Nistér. Bundle adjustment rules. In *Symp. of ISPRS Commision on Photogrammetric Computer Vision*, pages 266–271, Sep 2006.

[11] A. Eudes and M. Lhuillier. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[12] R. Eustice, H. Singh, J. Leonard, and M. Walter. Visually mapping the RMS Titanic: Conservative covariance estimates for SLAM information filters. *Intl. J. of Robotics Research*, 25(12):1223–1242, Dec 2006.

[13] G. H. Golub and R. J. Plemmons. Large-scale geodetic least-squares adjustment by dissection and orthogonal decomposition. *Linear Algebra Appl.*, 34:3–28, 1980.

[14] G. Guennebaud, B. Jacob, et al. Eigen v3. `http://eigen.tuxfamily.org`, 2010.

[15] P. J. Huber. *Robust statistics*. Springer Heidelberg, 2011.

[16] V. Ila, L. Polok, M. Šolony, P. Smrž, and P. Zemčík. Fast covariance recovery in incremental nonlinear least square solvers. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 4636–4643, May 2015.

[17] V. Ila, L. Polok, M. Šolony, and P. Svoboda. SLAM++-A highly efficient and temporally scalable incremental SLAM framework. *Intl. J. of Robotics Research*, Online First(0):1–21, 2017.

[18] V. Indelman, R. Roberts, C. Beall, and F. Dellaert. Incremental light bundle adjustment. In *British Machine Vision Conf. (BMVC)*, pages 134.1–134.11. BMVA Press, 2012.

[19] Y. Jeong, D. Nister, D. Steedly, R. Szeliski, and I.-S. Kweon. Pushing the envelope of modern methods for bundle adjustment. *IEEE Trans. Pattern Anal. Machine Intell.*, 34(8):1605–1617, 2012.

[20] Y.-D. Jian, D. Balcan, and F. Dellaert. Generalized subgraph preconditioners for large-scale bundle adjustment. In *Outdoor and Large-Scale Real-World Scene Analysis*, volume 7474 of *Lecture Notes in Computer Sci.*, pages 131–150. Springer Heidelberg, 2012.

[21] M. Kaess and F. Dellaert. Covariance recovery from a square root information matrix for data association. *Robotics and Autonomous Syst.*, 2009.

[22] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *Intl. J. of Robotics Research*, 31:217–236, Feb. 2011.

[23] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[24] M. I. Lourakis, A. Argyros, et al. Is Levenberg-Marquardt the most efficient optimization algorithm for implementing bundle adjustment? In *Intl. Conf. on Computer Vision (ICCV)*, volume 2, pages 1526–1531. IEEE, 2005.

[25] M. I. Lourakis and A. A. Argyros. SBA: A software package for generic sparse bundle adjustment. *ACM Trans. Math. Software*, 36(1):2, 2009.

[26] L. Moisan, P. Moulon, and P. Monasse. Automatic homographic registration of a pair of images, with a contrario elimination of outliers. *Image Processing On Line*, 2:56–73, 2012.

[27] P. Moulon, P. Monasse, R. Marlet, and Others. OpenMVG. `https://github.com/openMVG/openMVG`.

[28] R. Mur-Artal, J. Montiel, and J. D. Tardós. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robotics*, 31(5):1147–1163, 2015.

[29] 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.

[30] M. Pollefeys, M. Vergauwen, K. Cornelis, J. Tops, F. Verbiest, and L. V. Gool. Structure and motion from image sequences. In *Proc. Conf. on Optical 3-D Measurement Techniques*, pages 251–258. Vienna University of Technology, 2001.

[31] L. Polok, V. Ila, and P. Smrž. Fast sparse matrix multiplication on GPU. In *Proc. of the High Performance Computing Symp.* ACM, 2015.

[32] L. Polok, V. Ila, M. Šolony, P. Smrž, and P. Zemčík. Incremental block Cholesky factorization for nonlinear least squares in robotics. In *Robotics: Science and Systems (RSS)*, 2013.

[33] L. Polok, V. Lui, V. Ila, T. Drummond, and R. Mahony. The effect of different parameterisations in incremental structure from motion. In *Australian Conf. on Robotics and Automation (ACRA)*, December 2015.

[34] L. Polok, M. Šolony, V. Ila, P. Zemčík, and P. Smrž. Efficient implementation for block matrix operations for nonlinear least squares problems in robotic applications. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2013.

[35] M. J. Powell. A hybrid method for nonlinear equations. *Numerical methods for nonlinear algebraic equations*, 7:87–114, 1970.

[36] D. Rosen, M. Kaess, and J. Leonard. An incremental trust-region method for robust online sparse least-squares estimation. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1262–1269, St. Paul, MN, May 2012.

[37] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman. The new college vision and laser data set. *Intl. J. of Robotics Research*, 28(5):595–599, May 2009.

[38] 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.

[39] H. Strasdat. *Local Accuracy and Global Consistency for Efficient Visual SLAM*. PhD thesis, Imperial College London, UK, 2012.

[40] C. Strecha, W. von Hansen, L. Van Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2008.

[41] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Oct. 2012.

[42] N. Sünderhauf and P. Protzel. Switchable constraints for robust pose graph SLAM. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1879–1884. IEEE, 2012.

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

[44] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment – a modern synthesis. In *Vision Algorithms: Theory and Practice*, pages 298–372. Springer Heidelberg, 1999.