

3D Reconstruction Quality Analysis and Its Acceleration on GPU Clusters

Lukas Polok[‡], Viorela Ila^{*} and Pavel Smrz[‡]

[‡]Brno University of Technology, Faculty of Information Technology, IT4Innovations Centre of Excellence

^{*}Australian National University, ACR Centre of Excellence for Robotic Vision

Email: {ipolok, smrz}@fit.vutbr.cz, viorela.ila@anu.edu.au

Abstract—3D reconstruction has a wide variety of applications in computer graphics, robotics or digital cinema production, among others. With the rapid increase in computing power, it has become more feasible for the reconstruction algorithms to run online, even on mobile devices. Maximum likelihood estimation (MLE) is the adopted technique to deal with the sensor uncertainty. Most of the existing 3D reconstruction frameworks only recover the *mean* of the reconstructed geometry. Recovering also the *variance* is highly computationally intensive and is seldom performed. However, variance is the natural choice of estimate quality indicator. In this paper, the associated costs are analyzed and efficient but exact solutions to calculating partial matrix inverses are proposed, which apply to *any general* problem with many mutually independent variables. Speedups exceeding an order of magnitude are reported.

I. INTRODUCTION

A typical 3D reconstruction pipeline consists of several stages. First, visual features [1] are extracted from the images which are then matched using approximate nearest neighbor search [2] and subsequently pruned using RANSAC along with geometric estimation [3]. Depending on the scale of the problem, the matching can be either done in all-to-all manner or hierarchically using approximate clustering first and then fine grained matching within the clusters [4]. The *camera poses* are given by relative transformations between the matched images and the *landmark positions* are given by triangulation of the matched feature points. Because of different sources of the errors, the initial estimate alone is rather noisy and simply concatenating the calculated geometrical transformations and triangulating the observed feature points as they come, would quickly diverge catastrophically. Therefore, one more crucial step is employed: the bundle adjustment (BA).

Most of the 3D reconstruction implementations work incrementally: only one or a few frames are integrated at a time, followed by a BA step. Bundle adjustment finds the maximum likelihood estimation of the camera poses and of the structure, given the observations, and is most commonly solved using nonlinear least squares optimization. Other approaches [5] are possible. To solve the nonlinear least squares, conjugate gradient (CG) or a direct solver can be employed. While CG is often claimed as a linear cost algorithm when applied to linear solving [6], [7], it typically takes more iterations of the nonlinear solver to converge – being effectively slower than a direct solver.

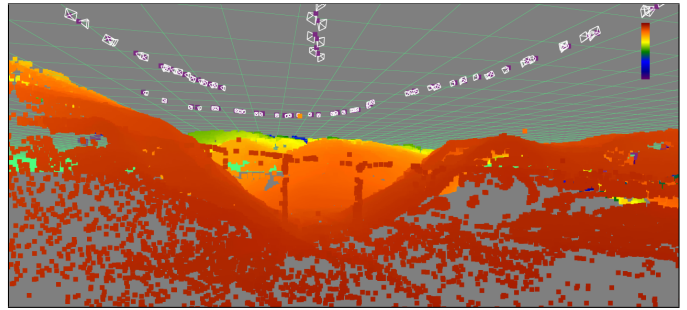


Fig. 1. Marginal covariances for the 3D reconstruction of the bridge sequence from the Fast & Furious 6 dataset, displayed in false colors (orange means high confidence, blue – low confidence). Data courtesy of Double Negative Visual Effects.

The seminal paper [8], [9] describes design and implementation of an efficient BA package with the basic traits shared by most of the other implementations. The problem is formulated as a Levenberg-Marquardt [10] nonlinear least squares optimization. It makes use of the problem sparsity: not all of the points are observed by all of the cameras and the system graph is usually far from being fully connected. It also makes use of the characteristics of the BA problem which typically contains a relatively large amount of *landmarks* that have no relations among themselves (they form a large independent set from the graphical point of view). This gives rise to diagonal sub-matrices that make the underlying linear problem easier to solve than by applying a general linear solver directly to the whole matrix.

Using unstructured photograph collections from the Internet allows for extremely large scale 3D reconstruction [4]. The problems that need to be tackled exist both in the vision part of the reconstruction pipeline as well as in the BA optimizer. In the vision part, the feature matching becomes the bottleneck, as it scales with $O(n^2)$ in the number of images and hierarchical matching is proposed to solve the problem both more efficiently and in parallel. The authors implement two optimization strategies which are selected based on problem size. The first one is a block diagonal preconditioned conjugate gradient solver. The second one is rather similar to a sparse BA (SBA), with the difference that unlike in SBA where the Schur complement [11] is solved using dense LDL^T factorization, a sparse Cholesky factorization is employed here to gain up to an order of magnitude speedup for large systems. Similar speedups were reproduced by e.g. [12].

To further accelerate the optimization part, it is possible to employ parallelization. CG solvers are parallelized easily, as they basically only require parallel implementation of sparse matrix-vector multiplication routine [13]. For direct solvers, distributed optimization techniques were proposed [14] where the problem is split into several sub-problems with minimal graph separators that are solved independently, followed by a separator optimization pass. Such methods can be easily used for parallelization on clusters.

Parameterizations taking advantage of the *incremental* solving were proposed as well, in [15], relative camera and pose formulation is employed, rather than using a single global Euclidean coordinate frame. After adding a new camera pose and the associated observations, it is possible to find the variables where this addition induced a significant change and only a reduced system consisting of those variables and their neighbors is solved. The size of the system that needs to be solved is only a fraction of the full system, making the optimization faster. A standard Schur complement solver is employed.

Another approach is acceleration via graph sparsification. In [16], rather than optimizing the entire problem only the camera poses are optimized, with the observations taking form of three-view constraints related to the tri-focal tensor. A similar generalized approach is proposed in [17] where the structure variables would be represented implicitly by the corresponding triangulation functions and therefore only the camera poses and optionally also their calibrations would be optimized. In both cases, the structure points can be triangulated after-the-fact in the least squares fashion from all the cameras that observe each given point. Since these methods effectively solve a pose graph, it is possible to use the appropriate incremental algorithms [18], [19], [20] as well.

The rest of the paper is structured as follows: in the next section, the related work is described. In section III, the nonlinear least squares is briefly revisited and a solver using Schur complement is formulated. section IV derives the formulas for efficient sparse covariance recovery from Schur complement. The performance of the proposed algorithm is evaluated in section V. The paper concludes with section VI.

II. RELATED WORK

Even though recovering the mean of the estimate in the BA problems is relatively simple even at large scale, as documented by the previous section, recovering its covariance is significantly more difficult. One of the problems is that while the system matrix is sparse and can get very large, its inverse is completely dense and the memory footprint of maintaining such a matrix would be prohibitive, easily reaching hundreds of GB. Fortunately, for quality assurance and many other applications, only certain parts of the inverse are of interest – especially its block diagonal. Still, the problem of the computational complexity remains, which is the likely reason this problem was not widely addressed before.

While the covariances are explicit in Kalman filters and can also be easily recovered in information filters, filtering is

not widely used in BA or the 3D reconstruction problems in general, since it is less efficient [21] and cannot take advantage of the various sparsity optimizations described in the previous section. In addition, there is a strict limit of the system size for which a dense matrix can be kept in RAM of today's systems, which would only allow moderate size 3D reconstructions.

Recovering the covariances in the context of nonlinear least squares is more difficult. Thrun [22] proposes the use of so-called Markov blankets to approximate covariances of the poses of a robot. These are sub-blocks in the inverses of smaller matrices that each contains the pose in question and also the adjacent landmarks. It has been shown that those estimates are over-confident.

In [7] a visual mapping of the sunken RMS Titanic is discussed and both the estimate and its covariance is recovered. The covariance is maintained incrementally: first, the covariances of the newly introduced variables are calculated by solving for the corresponding columns of the inverse system matrix. This column matrix is then fed to a bank of Kalman filters which update the covariances of the older variables.

An exact method for sparse covariance recovery was proposed in [23]. It is based on a recursive formula [24], [25], which calculates any covariance elements on demand from other covariance elements and elements of the Cholesky factorization of the system matrix. The downside of these methods is the need to calculate Cholesky factorization of the entire system, rather than to reuse the Schur complement and its factorization.

The authors of [26] compare the results of their reconstruction to a surveyed ground truth and employ the point to the nearest-ground-truth-triangle distance as the error metric. Here, the robustness of the results only depends on the point could alignment algorithm, which is a widely researched topic. While this is a viable alternative to covariances as a quality assurance method, it is of lower importance in real applications: why use a lower precision reconstruction if a precise one is at hand?

In the context of the technique proposed in this paper, it is not significant whether the underlying problem is monocular, stereo or other form of BA. It applies to *all* problems which can be solved efficiently using the standard Schur complement technique. Covariances in pose graph problems can be calculated efficiently using the approach described in our previous paper [27].

III. OPTIMIZATION PROBLEM FORMULATION

In our context, the estimation problem is formulated as a maximum likelihood estimation (MLE) of a set of variables $\theta = [\theta_1 \dots \theta_n]$ given a set of observations $\mathbf{z} = [z_1 \dots z_m]$. Without the loss of generality, it is possible to order the variables in such a way that $\theta_1 \dots \theta_p$ are the p camera poses and $\theta_{p+1} \dots \theta_{n=p+1+l}$ are the l landmark positions and to assume that each constraint is between a pose variable and a landmark variable. Situations with additional types of variables (e.g. the intrinsic camera parameters) are possible. Situations with only a single type of variable (e.g. as in pose graph

optimization) are also possible, although the ordering for Schur complement is more elaborate; one needs to compute the bipartite coloring if one exists or resort to maximum independent set if it does not.

The goal is to obtain the maximum likelihood estimate of a set of variables in θ , given the available observations in \mathbf{z} :

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

For every observation $z_k = h_k(\theta_{i_k}, \theta_{j_k}) - v_k$, the noise v_k is assumed to be normally distributed, with covariance Σ_k :

$$P(z_k | \theta_{i_k}, \theta_{j_k}) \propto \exp\left(-\frac{1}{2} \|h_k(\theta_{i_k}, \theta_{j_k}) \ominus z_k\|_{\Sigma_k}^2\right), \quad (2)$$

where $h_k(\theta_{i_k}, \theta_{j_k})$ is the nonlinear measurement function, z_k is the constraint between the variables θ_{i_k} and θ_{j_k} , \ominus is the vectorial inverse composition operator. The MLE in (1) is found by solving the following nonlinear least squares (NLS):

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \left\{ \frac{1}{2} \sum_{k=1}^m \|h_k(\theta_{i_k}, \theta_{j_k}) \ominus z_k\|_{\Sigma_k}^2 \right\}. \quad (3)$$

Iterative methods, such as Gauss-Newton are often used to solve this NLS. This is usually addressed by solving a sequence of linear systems at every iteration. Linear approximations of the nonlinear residual functions around the current *linearization point* θ^i are calculated as:

$$\tilde{\mathbf{r}}(\theta^i) = \mathbf{r}(\theta^i) + J(\theta^i)(\theta \ominus \theta^i), \quad (4)$$

with $\mathbf{r}(\theta) = [r_1, \dots, r_m]^\top$ being a vector gathering all nonlinear residuals of the type $r_k = h_k(\theta_{i_k}, \theta_{j_k}) \ominus z_k$ and J being the Jacobian matrix which gathers the derivatives of the components of $\mathbf{r}(\theta)$. With this, the NLS in (3) is approximated by a linear one and solved by successive iterations:

$$\delta^* = \underset{\delta}{\operatorname{argmin}} \frac{1}{2} \|A\delta - \mathbf{b}\|^2, \quad (5)$$

with the matrix $A \triangleq \Sigma^{-1/2} J$ and the vector $\mathbf{b} \triangleq -\Sigma^{-1/2} \mathbf{r}$ defined the same as in [28]. The correction $\delta \triangleq \theta \ominus \theta^i$ towards the solution is obtained by solving the *normal equation*:

$$A^\top A \delta = A^\top \mathbf{b}, \text{ or } \Lambda \delta = \boldsymbol{\eta}, \quad (6)$$

where we define the *information matrix* $\Lambda \triangleq A^\top A$ and the right hand side (r.h.s) $\boldsymbol{\eta} \triangleq A^\top \mathbf{b}$.

By taking advantage of the structure of the problem, rather than solving the normal equation directly using sparse factorization solver, it is possible to employ the Schur complement trick. In case the poses are ordered first, followed by all the landmarks, the normal equation (6) can be partitioned as:

$$\begin{bmatrix} A & U \\ U^\top & D \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix}, \quad (7)$$

where the D is supposed to be invertible and also block diagonal (since there are no observations that would relate two landmark variables and therefore no off-diagonal blocks are filled). The Schur complement of A is:

$$\text{Schur}(A) \triangleq A - U D^{-1} U^\top. \quad (8)$$

This can be used to solve for the original system as:

$$(A - U D^{-1} U^\top) x = a - U D^{-1} b \quad (9)$$

$$y = D^{-1} (b - U^\top x), \quad (10)$$

where the former is a smaller, more dense system that can be solved using a general linear solver and the latter is merely a matrix-vector product. The advantage of this procedure is that inverting D amounts to inverting its individual diagonal blocks which is embarrassingly parallel operation. Additionally, in the BA problems D contains the most of the rank of the system matrix so that the large part of the system is solved quickly.

To solve for (9), several types of direct solvers have been applied in literature. It is possible to use dense Cholesky or dense LDL^\top decompositions¹. Densities of as high as 40 % occur on e.g. the Venice dataset [29]. Sparse Cholesky solvers have shown about an order of magnitude speedups, especially on large systems and while using a good ordering. The fill-reducing orderings used for sparse Cholesky in BA implementations include MMD [30], AMD [31] or even reverse Cuthill-McKee [32] (although most likely only in an attempt to point at disadvantages of direct solvers). For perspective, dense Cholesky solver on GPU achieves about two orders of magnitude speedup (including data transfer) but is limited by the available memory.

While sparse LDL^\top , LU or even QR seem like viable options, it is required to take the pivoting into the account: these factorizations are not implicitly numerically stable (unlike Cholesky) and may require row or column interchanges as the factorization progresses. These interchanges are typically implemented to improve the results numerically but ignore the fill-in they cause. E.g. calculating the LU decomposition of the Λ matrix of Venice requires more than 15 GB of memory and two hours of runtime (while Cholesky runs in a few seconds).

IV. COMPUTING COVARIANCES EFFICIENTLY

The covariance of the estimated variables is obtained by taking an inverse of the information matrix, $\Sigma = \Lambda^{-1}$. Carrying this inverse out yields a dense matrix, as already mentioned above, and should be avoided. In case a sparse factorization $\Lambda = R^\top R$ can be calculated, the following recursive formula [24], [25] can be used to calculate elements of Σ at the positions of non-zero elements in R :

$$\Sigma_{i,i} = \frac{1}{R_{i,i}} \left(\frac{1}{R_{i,i}} - \sum_{k=i+1, R_{i,k} \neq 0}^n R_{i,k} \Sigma_{k,i} \right), \quad (11)$$

$$\Sigma_{i,j} = \frac{1}{R_{i,i}} \left(- \sum_{k=i+1, R_{i,k} \neq 0}^j R_{i,k} \Sigma_{k,j} - \sum_{k=j+1, R_{i,k} \neq 0}^n R_{i,k} \Sigma_{j,k} \right), \quad (12)$$

where $M_{i,j}$ refers to an element of some matrix M at row i and column j . Unfortunately, this formula cannot be directly applied to the Schur-complemented system $\Lambda \Sigma = I$:

$$\begin{bmatrix} A & U \\ U^\top & D \end{bmatrix} \cdot \begin{bmatrix} \Sigma_p & \Sigma_{pl} \\ \Sigma_{pl}^\top & \Sigma_l \end{bmatrix} = \begin{bmatrix} I_p & 0 \\ 0^\top & I_l \end{bmatrix}. \quad (13)$$

¹In here, the D is a generic diagonal matrix, other than that in (8).

where both Σ and the identity matrix I are partitioned the same way as Λ is partitioned in (7). Note that the subscripts here are only identifiers rather than element indices. By taking Cholesky decomposition $S^\top S \triangleq \text{Schur}(A)$, the covariances of the camera variables are:

$$S^\top S \Sigma_p = I_p - U D^{-1} 0^\top = I_p \text{ so } \Sigma_p = (S^\top S)^{-1}, \quad (14)$$

and thus the recursive formula in (11) and (12) can be used efficiently.

The situation is more interesting in recovering the covariances of the landmarks. It would be possible to make use of $T^\top T \triangleq \text{Schur}(D)$ and (14) to write:

$$\Sigma_l = (D - U^\top A^{-1} U)^{-1} = (T^\top T)^{-1}. \quad (15)$$

The matrix inverted here is positive definite and the recursive formula could be used again. However, the inverse A^{-1} is involved here: unless the underlying problem forms a bipartite graph which only really happens with vanilla forms of BA. As soon as e.g. intrinsic camera parameters, GPS or odometry measurements are introduced, A is no longer block diagonal and inverting it is much more difficult than inverting D in (8). Applying the Woodbury formula to (15):

$$\Sigma_l = D^{-1} + D^{-1} U^\top (A - U D^{-1} U^\top)^{-1} U D^{-1}, \quad (16)$$

$$\Sigma_l = D^{-1} + D^{-1} U^\top \Sigma_p U D^{-1}, \quad (17)$$

$$\Sigma_l = D^{-1} + D^{-1} U^\top S^{-1} S^{-\top} U D^{-1}. \quad (18)$$

Evaluating all of (18) would yield a dense matrix with the size approaching that of the full Σ which would be counter-productive. Instead, taking advantage of symmetry of Λ (and thus also of D and Σ), it is possible to write $B \triangleq S^{-\top} U D^{-1}$ in order to get $\Sigma_{l_{i,j}} = D_{i,j}^{-1} + B_{i,*}^\top \cdot B_{*,j}$. Note that $U D^{-1}$ is a sparse matrix with the number of nonzero blocks in each column equal to the number of cameras that observe the point corresponding to that column; $S^{-\top}$ can be efficiently calculated using sparse sparse back-substitution.

Finally, to get the cross-covariances between the camera and the landmark variables, it is possible to use (9) with covariance in place of a and identity on the right:

$$\Sigma_{pl} = (S^\top S) \setminus (0 - U D^{-1} I_l), \quad (19)$$

$$\Sigma_{pl} = -\Sigma_p U D^{-1}. \quad (20)$$

Again, computation can be saved by taking advantage of sparsity of the matrices so that recovering the full Σ_p is not necessary.

V. EXPERIMENTAL EVALUATION

The proposed method for recovering marginal covariances of points was tested on two public datasets, the Guildford Cathedral², Venice [29] and on the Fast & Furious 6 dataset which was kindly provided by Double Negative Visual Effects³. Two additional methods were compared: recursive formula on Cholesky factor of the system matrix, and recursive

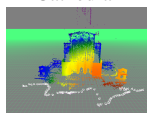
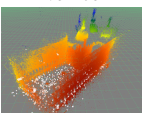
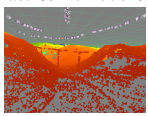
	Cathedral	Venice	Fast & Furious 6
			
Cameras	92	871	160
Landmarks	57, 957	530, 304	136, 453
Visibility	7.28 obs. / lm.	5.35 obs. / lm.	3.42 obs. / lm.
Λ	142.93 MB	980.33 MB	167.70 MB
Schur(A)	1.04 MB	45.06 MB	1.14 MB
S	1.04 MB	84.60 MB	1.73 MB

TABLE I
CHARACTERISTICS OF THE EVALUATED DATASETS.

formula on Schur(D) as in (15). More details about the datasets are listed in Table I.

The experiments were performed on the Salomon supercomputer, part of the IT4I Czech National Supercomputing Center. Each compute node is equipped with a pair of 12-core Xeon E5-2680 v3 running at 2.50 GHz and 128 GB of RAM. Memory consumption tests were performed on SGI UV2000 node, equipped with 14 of 8-core Xeon E5-4627 v2 at 3.3 GHz and 3.25 TB (Terabyte) of RAM; timing of these tests is denoted by the dagger[†] symbol.

Sparse block schemes [33] were used throughout the whole implementation, which previously proved about an order of magnitude speedups for batch recursive formula [27]. Block matrix products and decompositions were accelerated by Tesla K20x GPU.

	Cathedral	Venice	Fast & Furious 6
Proposed	0.165 s	7.060 s	0.293 s
Size of sparse $S^{-\top}$	1.24 MB	109.79 MB	2.97 MB
Chol(Λ)	1.251 s	16.856 s	0.951 s
Rec. formula all	3.308 s	82.689 s	3.493 s
Size of Chol(Λ)	74.05 MB	572.52 MB	93.33 MB
Schur(D)	160.662 s	4457.539 [†] s	149.999 s
Chol(Schur(D))	73 [†] hours	N/A	139 [†] hours
Rec. formula lm.	4459.647 [†] s	N/A	5 [†] hours
Size of Schur(D)	37.87 GB	398.01 GB	46.95 GB
Size of Chol(T)	106.70 GB	~ 7.37 TB	493.43 GB

TABLE II
TIMING RESULTS AND THE ASSOCIATED SPACE REQUIREMENTS OF THE EVALUATED METHODS (BEST TIMES IN BOLD).

Times required to calculate the marginal covariances are reported in Table II. The computation of the covariances of landmarks directly from Schur complement is the fastest for all tested datasets, followed by the use of recursive formula. The proposed method provides more than an order of magnitude speedup. The use of Schur(D) and recursive formula is prohibitive by both time and considerable memory requirements.

The magnitudes of the calculated landmark covariances are displayed as false color, see Figure 1 or Table I. From the colored view, it is apparent which parts of the reconstruction are more precise and which are not. The user can use this type of images to re-capture poorly reconstructed areas and obtain a high accuracy 3D reconstruction.

²can be obtained at <http://cvssp.org/impart/>

³<http://www.dneg.com/>

VI. CONCLUSION

We proposed methods for efficiently finding covariances in NLS problems which are solved using Schur complement, such as BA. The implementation of the proposed formulas significantly outperformed the existing methods, by a factor of $20\times$ for Cathedral, $12\times$ for Venice and $12\times$ for Fast & Furious 6. At the same time, the memory consumption for calculating the inverse of square root of the Schur complement is comparable to the storage of the square root itself (which is required by the nonlinear solver), and is much smaller than the storage needed for square rooting the full system for recursive formula. Using the Schur complement of the landmarks is prohibitive as it requires tens to hundreds of GB of storage.

The calculated covariances can then be interactively displayed using false color rendering and used for quality assessment of the 3D reconstruction. The proposed methods are fast enough to be run on-set so that additional data capture can take place if the reconstruction quality is not good enough.

ACKNOWLEDGMENTS

The research leading to these results has received funding from Horizon2020 project 644632-MixedEmotions, Czech Republic Technological Agency project TA144S01001 TraumaTech and the IT4IXS IT4Innovations Excellence in Science project (LQ1602).

We thank ARC Centre of Excellence for Robotic Vision, project number CE140100016 for funding Dr. V. Ila's research.

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPUs used for this research.

REFERENCES

- [1] D. G. Lowe, "Object recognition from local scale-invariant features," in *Intl. Conf. on Computer Vision (ICCV)*, vol. 2. IEEE, 1999, pp. 1150–1157.
- [2] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Proc. of the Intl. Conf. on Computer Vision Theory and Applications (VISAPP)*. INSTICC Press, 2009, pp. 331–340.
- [3] D. Nistér, "An efficient solution to the five-point relative pose problem," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 26, no. 6, pp. 756–770, 2004.
- [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] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment – a modern synthesis," in *Vision Algorithms: Theory and Practice*. Springer Heidelberg, 1999, pp. 298–372.
- [6] M. Byröd and K. Åström, "Conjugate gradient bundle adjustment," in *Eur. Conf. on Computer Vision (ECCV)*. Springer Heidelberg, 2010, pp. 114–127.
- [7] 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*, vol. 25, no. 12, pp. 1223–1242, Dec 2006.
- [8] M. Lourakis and A. Argyros, "The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm," Technical Report 340, Institute of Computer Science-FORTH, Heraklion, Crete, Greece, Tech. Rep., 2004.
- [9] M. I. Lourakis and A. A. Argyros, "SBA: A software package for generic sparse bundle adjustment," *ACM Trans. Math. Software*, vol. 36, no. 1, p. 2, 2009.
- [10] J. J. Moré, "The levenberg-marquardt algorithm: implementation and theory," in *Proc. of the Biennial Conf. on Numerical Analysis*. Springer Heidelberg, 1978, pp. 105–116.
- [11] F. Zhang, *The Schur complement and its applications*. Springer US, 2005, vol. 4.
- [12] K. Konolige, "Sparse sparse bundle adjustment," in *British Machine Vision Conf. (BMVC)*, Aberystwyth, Wales, 08/2010 2010.
- [13] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz, "Multicore bundle adjustment," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2011, pp. 3057–3064.
- [14] 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. [Online]. Available: <http://www.cc.gatech.edu/~dellaert/pubs/Ni07iccv.pdf>
- [15] D. Sibley, C. Mei, I. Reid, and P. Newman, "Adaptive relative bundle adjustment," in *Robotics: Science and Systems (RSS)*, vol. 32, 2009, p. 33.
- [16] V. Indelman, R. Roberts, C. Beall, and F. Dellaert, "Incremental light bundle adjustment," in *British Machine Vision Conf. (BMVC)*. BMVA Press, 2012, pp. 134.1–134.11.
- [17] L. Carlone, Z. Kira, C. Beall, V. Indelman, and F. Dellaert, "Eliminating conditionally independent sets in factor graphs: A unifying perspective based on smart factors," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 4290–4297.
- [18] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Trans. Robotics*, vol. 24, no. 6, pp. 1365–1378, Dec 2008.
- [19] 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*, vol. 31, pp. 217–236, Feb. 2011.
- [20] L. Polok, V. Ila, M. Šolony, P. Smrž, and P. Zemčik, "Incremental block Cholesky factorization for nonlinear least squares in robotics," in *Robotics: Science and Systems (RSS)*, 2013.
- [21] H. Strasdat, J. M. Montiel, and A. J. Davison, "Visual SLAM: why filter?" *Image and Vision Computing*, vol. 30, no. 2, pp. 65–77, 2012.
- [22] 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*, vol. 23, no. 7-8, pp. 693–716, 2004.
- [23] M. Kaess and F. Dellaert, "Covariance recovery from a square root information matrix for data association," *Robotics and Autonomous Systems*, 2009.
- [24] A. Björck, *Numerical methods for least squares problems*. SIAM, 1996.
- [25] G. H. Golub and R. J. Plemmons, "Large-scale geodetic least-squares adjustment by dissection and orthogonal decomposition," *Linear Algebra Appl.*, vol. 34, pp. 3–28, 1980.
- [26] M. Pollefeys, D. Nistér, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.-J. Kim, P. Merrell et al., "Detailed real-time urban 3D reconstruction from video," *Intl. J. of Computer Vision*, vol. 78, no. 2–3, pp. 143–167, 2008.
- [27] V. Ila, L. Polok, M. Šolony, P. Smrž, and P. Zemčik, "Fast covariance recovery in incremental nonlinear least square solvers," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2015, pp. 4636–4643.
- [28] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," *Intl. J. of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, Dec 2006.
- [29] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.
- [30] J. W. Liu, "Modification of the minimum-degree algorithm by multiple elimination," *ACM Trans. Math. Software*, vol. 11, no. 2, pp. 141–153, 1985.
- [31] P. R. Amestoy, T. A. Davis, and I. S. Duff, "An approximate minimum degree ordering algorithm," *SIAM J. on Matrix Analysis and Applications*, vol. 17, no. 4, pp. 886–905, 1996.
- [32] E. Cuthill, "Several strategies for reducing the bandwidth of matrices," in *Sparse Matrices and Their Applications*. Springer US, 1972, pp. 157–166.
- [33] L. Polok, V. Ila, and P. Smrž, "Cache efficient implementation for block matrix operations," in *Proc. of the High Performance Computing Symp.* ACM, 2013, pp. 698–706.