Square Root Bundle Adjustment for Large-Scale Reconstruction

Nikolaus Demmel   Christiane Sommer   Daniel Cremers   Vladyslav Usenko
Technical University of Munich
{nikolaus.demmel,c.sommer,cremers,vlad.usenko}@tum.de

Abstract

We propose a new formulation for the bundle adjustment problem which relies on nullspace marginalization of landmark variables by QR decomposition. Our approach, which we call square root bundle adjustment, is algebraically equivalent to the commonly used Schur complement trick, improves the numeric stability of computations, and allows for solving large-scale bundle adjustment problems with single-precision floating-point numbers. We show in real-world experiments with the BAL datasets that even in single precision the proposed solver achieves on average equally accurate solutions compared to Schur complement solvers using double precision. It runs significantly faster, but can require larger amounts of memory on dense problems. The proposed formulation relies on simple linear algebra operations and opens the way for efficient implementations of bundle adjustment on hardware platforms optimized for single-precision linear algebra processing.

1. Introduction

Bundle adjustment (BA) is a core component of many 3D reconstruction algorithms and structure-from-motion (SfM) pipelines. It is one of the classical computer vision problems and has been investigated by researchers for more than six decades [7]. While different formulations exist, the underlying question is always the same: given a set of approximate point (landmark) positions that are observed from a number of cameras in different poses, what are the actual landmark positions and camera poses? One can already compute accurate 3D positions with only a few images; however, with more available images we will get a more complete reconstruction. With the emergence of large-scale internet photo collections has come the need to solve bundle adjustment on a large scale, i.e., for thousands of images and hundreds of thousands of landmarks. This
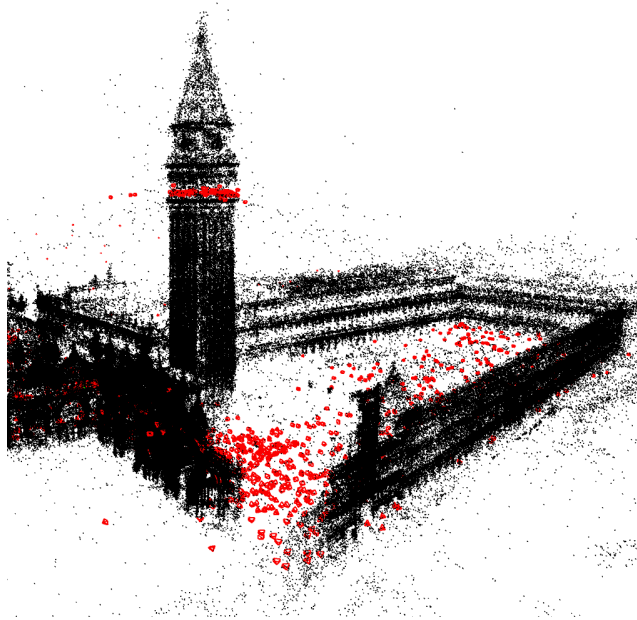
Figure 1: Optimized 3D reconstruction of the largest *venice* BAL dataset with 1778 cameras, around one million landmarks, and five million observations. For this problem, the proposed square root bundle adjustment ($\sqrt{BA}$) solver is 42% faster than the best competing method at reaching a cost tolerance of 1%.

requires scalable solutions that are still efficient for large problems and do not run into memory or runtime issues.

Modern BA solvers usually rely on the Schur complement (SC) trick that computes the normal equations of the original BA problem and then breaks them down into two smaller subproblems, namely (1) the solution for camera poses and (2) finding optimal landmark positions. This results in the drastically smaller reduced camera system (RCS), which is also better-conditioned [2] than the original normal equations. The use of the Schur complement has become the de facto standard for solving large-scale bundle adjustment and is hardly ever questioned.

In this work, we challenge the widely accepted assump-

tion of SC being the best choice for solving bundle adjustment, and provide a detailed derivation and analysis of an alternative problem reduction technique based on QR decomposition. Inspired by similar approaches in the Kalman filter literature [34], we factorize the landmark Jacobian J_l such that we can project the original problem onto the nullspace of J_l. Thus, we circumvent the computation of the normal equations and their system matrix H, and are able to directly compute a matrix square root of the RCS while still solving an algebraically equivalent problem. This improves numerical stability of the reduction step, which is of particular importance on hardware optimized for single-precision floats. Following terminology for nullspace marginalization on Kalman filters, we call our method *square root bundle adjustment*, short $\sqrt{BA}$. In particular, our contributions are as follows:

- We propose nullspace marginalization as an alternative to the traditional Schur complement trick and prove that it is algebraically equivalent.
- We closely link the very general theory of nullspace marginalization to an efficient implementation strategy that maximally exploits the specific structure of bundle adjustment problems.
- We show that the algorithm is well parallelizable and that the favorable numerical properties admit computation in single precision, resulting in an additional twofold speedup.
- We perform extensive evaluation of the proposed approach on the Bundle Adjustment in the Large (BAL) datasets and compare to the state-of-the-art optimization framework Ceres.
- We release our implementation and complete evaluation pipeline as open source to make our experiments reproducible and facilitate further research:
  `https://go.vision.in.tum.de/rootba`.

2. Related work

We propose a way to solve large-scale bundle adjustment using QR decomposition, so we review both works on bundle adjustment (with a focus on large-scale problems) and works that use QR decomposition for other tasks in computer vision and robotics. For a general introduction to numerical algorithms (including QR decomposition and iterative methods for solving linear systems), we refer to [5, 15].

(Large-scale) bundle adjustment A detailed overview of bundle adjustment in general can be found in [31], including an explanation of the Schur complement (SC) reduction technique [7] to marginalize landmark variables. Byröd and Åström use the method of conjugate gradients (CG) on the normal equations [17, 6] to minimize the linearized least squares problem without the Schur complement [8]. They

also QR-decompose the Jacobian, but only for block preconditioning without marginalizing landmarks. Agarwal et al. have proposed preconditioned CG on the RCS after SC to solve the large-scale case [2], and Wu et al. further extend these ideas to a formulation which avoids explicit computation of the SC matrix [32]. A whole number of other works have proposed ways to further improve efficiency, accuracy and/or robustness of BA [13, 21, 20, 35, 27], all of them using the Schur complement. More recently, in Stochastic BA [36] the reduced system matrix is further decomposed into subproblems to improve scalability. Several open source BA implementations are available, e.g., the SBA package [24], the g2o framework [22], or Ceres Solver [1], which has become a standard tool for solving BA-like problems in both academia and industry.

Nullspace marginalization, square root filters, and QR decomposition The concept of nullspace marginalization has been used in contexts other than BA, e.g., for the multi-state constraint Kalman filter [26] and earlier in [3]. [34] proves the equivalence of nullspace marginalization and the Schur complement in the specific case of robot SLAM. Several works explicitly point out the advantage of matrix square roots in state estimation [25, 4, 11, 33], but to the best of our knowledge matrix square roots have not yet been used for problem size reduction in BA. The QRkit [30] emphasizes the benefits of QR decomposition for sparse problems in general and also mentions BA as a possible application, but the very specific structure of BA problems and the involved matrices is not addressed. The orthogonal projector used in the Variable Projection (VarPro) method [28, 18] is related to the nullspace marginalization in our approach. However, VarPro focuses on separable non-linear least squares problems, which do not include standard BA. While [18] mentions the use of QR decomposition to improve numeric stability, we take it one step further by more efficiently multiplying in-place with $Q_2^\top$ rather than explicitly with $I - Q_1 Q_1^\top$ (further discussed in Section 4.3). This also enables our very efficient way to compute landmark damping (not used in VarPro). Our landmark blocks can be seen as a specific instance of Smart Factors proposed in [10], where nullspace projection with explicit SVD decomposition is considered. Instead of the re-triangulation in [10], we suggest updating the landmarks with back substitution. The factor grouping in [9] allows to mix explicit and implicit SC. This idea is orthogonal to our approach and could be considered in future work.

3. QR decomposition

We briefly introduce the QR decomposition, which can be computed using Givens rotations (see appendix). For further background, we refer the reader to textbooks on least squares problems (e.g., [5]). Let A be an $m \times n$ matrix

of full rank with $m \geq n$, i.e., $\text{rank}(A) = n$. A can be decomposed into an $m \times m$ orthogonal matrix Q and an $m \times n$ upper triangular matrix R. As the bottom $(m - n)$ rows of R are zeros, we can partition R and Q:

$$A = QR = Q \begin{pmatrix} R_1 \\ 0 \end{pmatrix} = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \begin{pmatrix} R_1 \\ 0 \end{pmatrix} = Q_1 R_1 \,, \quad (1)$$

where R_1 is an $n \times n$ upper triangular matrix, Q_1 is $m \times n$, and Q_2 is $m \times (m - n)$. Note that this partitioning of Q directly implies that the columns of Q_2 form the left nullspace of A, i.e., $Q_2^\top A = 0$. Since Q is orthogonal, we have

$$Q^\top Q = \mathbf{I}_m = QQ^\top \,, \quad (2)$$

where $\mathbf{I}_m$ is the $m \times m$ identity matrix. From (2) we derive:

$$Q_1^\top Q_1 = \mathbf{I}_n \,, \quad Q_2^\top Q_2 = \mathbf{I}_{m-n} \,, \quad Q_1^\top Q_2 = 0 \,, \quad (3)$$

$$Q_1 Q_1^\top = \mathbf{I}_m - Q_2 Q_2^\top \,. \quad (4)$$

4. Square root bundle adjustment

We assume a very general form of bundle adjustment, similar to [2]: let x be a state vector containing all the optimization variables. We can subdivide x into a pose part x_p containing extrinsic and possibly intrinsic camera parameters for all n_p images (index i), and a landmark part x_l consisting of the 3D coordinates of all n_l landmarks (index j). The total bundle adjustment energy is a sum of squared residuals

$$E(x_p, x_l) = \sum_i \sum_{j \in O(i)} r_{ij}(x_p, x_l)^2 = \|r(x_p, x_l)\|^2 \,, \quad (5)$$

where $j \in O(i)$ means that landmark j is observed in frame i and $r(x)$ is the concatenation of all residuals r_{ij} into one vector. We call the total number of residuals N_r. For a pose dimensionality d_p, the length of the total state vector (x_p, x_l) is $d_p n_p + 3n_l =: N_p + 3n_l$. Typically, $d_p = 6$ if only extrinsic camera parameters need to be estimated, and $d_p = 9$ if intrinsic calibration is also unknown.

4.1. Least squares problem

The energy in (5) is usually minimized by the Levenberg-Marquardt algorithm, which is based on linearizing $r(x)$ around the current state estimate $x^0 = (x_p^0, x_l^0)$ and then solving a damped linearized problem

$$\min_{\Delta x} \left\| \begin{pmatrix} r \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} J_p & J_l \\ \sqrt{\lambda} D_p & 0 \\ 0 & \sqrt{\lambda} D_l \end{pmatrix} \begin{pmatrix} \Delta x_p \\ \Delta x_l \end{pmatrix} \right\|^2 =$$

$$\min_{\Delta x_p, \Delta x_l} \left(\left\| r + \begin{pmatrix} J_p & J_l \end{pmatrix} \begin{pmatrix} \Delta x_p \\ \Delta x_l \end{pmatrix} \right\|^2 \quad (6) \right.$$

$$\left. + \lambda \| D_p \Delta x_p \|^2 + \lambda \| D_l \Delta x_l \|^2 \right) ,$$

with $r = r(x^0)$, $J_p = \left. \frac{\partial r}{\partial x_p} \right|_{x^0}$, $J_l = \left. \frac{\partial r}{\partial x_l} \right|_{x^0}$, and $\Delta x = x - x^0$. Here, λ is a damping coefficient and D_p and D_l are diagonal damping matrices for pose and landmark variables (often $D^2 = \text{diag}(J^\top J)$).

To simplify notation, in this section we consider the undamped problem (i.e., $\lambda = 0$) and discuss the efficient application of damping in Section 5.3. The undamped problem in (6) can be solved by forming the corresponding normal equation

$$\begin{pmatrix} H_{pp} & H_{pl} \\ H_{lp} & H_{ll} \end{pmatrix} \begin{pmatrix} -\Delta x_p \\ -\Delta x_l \end{pmatrix} = \begin{pmatrix} b_p \\ b_l \end{pmatrix} , \quad (7)$$

where

$$H_{pp} = J_p^\top J_p \,, \quad H_{ll} = J_l^\top J_l \,, \quad (8)$$

$$H_{pl} = J_p^\top J_l = H_{lp}^\top \,, \quad (9)$$

$$b_p = J_p^\top r \,, \quad b_l = J_l^\top r \,. \quad (10)$$

The system matrix H of this problem is of size $(N_p + 3n_l)^2$, which can become impractically large (millions of rows and columns) for problems like those in [2] (see Figure 1 for an example).

4.2. Schur complement (SC)

A very common way to solve (7) is by applying the Schur complement trick (see e.g., [7, 2, 32]): we form the RCS

$$\tilde{H}_{pp}(-\Delta x_p) = \tilde{b}_p \,, \quad (11)$$

with

$$\tilde{H}_{pp} = H_{pp} - H_{pl} H_{ll}^{-1} H_{lp} \,, \quad (12)$$

$$\tilde{b}_p = b_p - H_{pl} H_{ll}^{-1} b_l \,. \quad (13)$$

The solution Δx_p^* of (11) is the same as the pose component of the solution of (7), but now the system matrix has a much more tractable size of N_p^2, which is usually in the order of thousands $\times$ thousands. Note that as H_{ll} is block-diagonal with blocks of size 3×3, the multiplication with H_{ll}^{-1} is cheap. Given an optimal pose update Δx_p^*, the optimal landmark update is found by back substitution

$$-\Delta x_l^* = H_{ll}^{-1}(b_l - H_{lp}(-\Delta x_p^*)) \,. \quad (14)$$

4.3. Nullspace marginalization (NM)

Using QR decomposition on $J_l = QR$, and the invariance of the L2 norm under orthogonal transformations, we can rewrite the term in (6):

$$\| r + \begin{pmatrix} J_p & J_l \end{pmatrix} \begin{pmatrix} \Delta x_p \\ \Delta x_l \end{pmatrix} \|^2$$

$$= \| Q^\top r + \begin{pmatrix} Q^\top J_p & Q^\top J_l \end{pmatrix} \begin{pmatrix} \Delta x_p \\ \Delta x_l \end{pmatrix} \|^2 \quad (15)$$

$$= \| Q_1^\top r + Q_1^\top J_p \Delta x_p + R_1 \Delta x_l \|^2$$

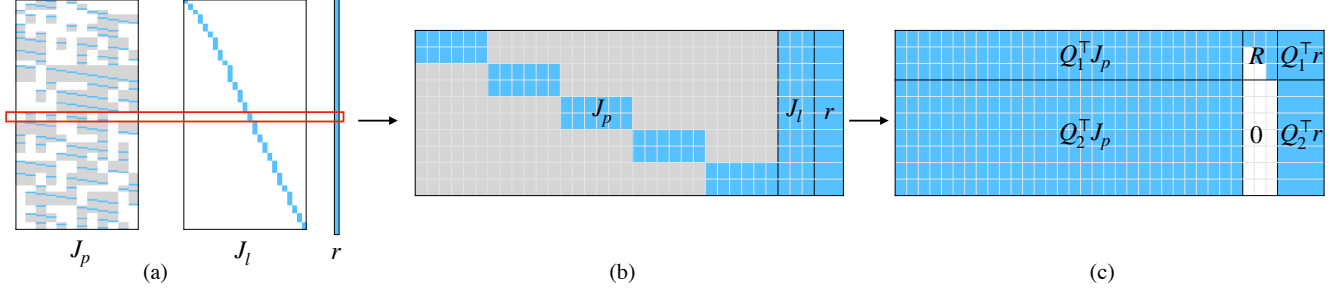$$\quad + \| Q_2^\top r + Q_2^\top J_p \Delta x_p \|^2 \,.$$

Figure 2: Dense landmark blocks. (a) Sparsity structure of the pose Jacobian is fixed during the optimization. Non-zero elements shown in blue, potentially non-zero elements after Givens QR shown in gray, and elements that will always stay zero shown in white. (b) Dense storage for the outlined (red) landmark block that efficiently stores all Jacobians and residuals for a single landmark. (c) Same landmark block after in-place marginalization. As Givens rotations operate on individual rows, marginalization can be performed for each landmark block separately, possibly in parallel.

As R_1 is invertible, for a given Δx_p^*, the first term can always be set to zero (and thus minimized) by choosing

$$\Delta x_l^* = -R_1^{-1}(Q_1^\top r + Q_1^\top J_p \Delta x_p^*). \tag{16}$$

So (6) reduces to minimizing the second term in (15):

$$\min_{\Delta x_p} \|Q_2^\top r + Q_2^\top J_p \Delta x_p\|^2. \tag{17}$$

Again, this problem is of significantly smaller size than the original one. However, as opposed to the (explicit) Schur complement trick, we do not explicitly have to form the Hessian matrix.

4.4. Equivalence of SC and NM

With the QR decomposition $J_l = Q_1 R_1$ used in the last paragraphs, we get

$$H_{pp} = J_p^\top J_p, \tag{18}$$
$$H_{pl} = J_p^\top Q_1 R_1, \tag{19}$$
$$H_{ll} = R_1^\top Q_1^\top Q_1 R_1 = R_1^\top R_1, \tag{20}$$
$$b_p = J_p^\top r, \tag{21}$$
$$b_l = R_1^\top Q_1^\top r. \tag{22}$$

Using this, we can rewrite the Schur complement matrix $\tilde{H}_{pp}$ and vector $\tilde{b}_p$ and simplify with (4):

$$\tilde{H}_{pp} = H_{pp} - J_p^\top Q_1 R_1 (R_1^\top R_1)^{-1} R_1^\top Q_1^\top J_p$$
$$= H_{pp} - J_p^\top (\mathbf{I}_{N_r} - Q_2 Q_2^\top) J_p \tag{23}$$
$$= J_p^\top Q_2 Q_2^\top J_p,$$
$$\tilde{b}_p = b_p - J_p^\top Q_1 R_1 (R_1^\top R_1)^{-1} R_1^\top Q_1^\top r$$
$$= J_p^\top Q_2 Q_2^\top r. \tag{24}$$

Thus, the SC-reduced equation is nothing but the normal equation of problem (17), which proves the algebraic equivalence of the two marginalization techniques. Additionally,

we can show that the equations for back substitution for Schur complement (14) and nullspace marginalization (16) are also algebraically equivalent:

$$\Delta x_l^* = -H_{ll}^{-1}(b_l + H_{lp} \Delta x_p^*)$$
$$= -(R_1^\top R_1)^{-1}(R_1^\top Q_1^\top r + (J_p^\top Q_1 R_1)^\top \Delta x_p^*) \tag{25}$$
$$= -R_1^{-1}(Q_1^\top r + Q_1^\top J_p \Delta x_p^*).$$

Note that the above arguments also hold for the damped problem (6), the difference being that the Hessian will have an augmented diagonal and that the matrices in the QR decomposition will have a larger size.

5. Implementation details

Bundle adjustment is a very structured problem, so we can take advantage of the problem-specific matrix structures to enable fast and memory-efficient computation.

5.1. Storage

We group the residuals by landmarks, such that J_l has block-sparse structure, where each block is $2k_j \times 3$ with k_j the number of observations for a particular landmark, see Figure 2 (a). As each landmark is only observed by a subset of cameras, the pose Jacobian J_p is also sparse.

We group the rows corresponding to each landmark and store them in a separate dense memory block, which we name a *landmark block*. We store only the blocks of the pose Jacobian that correspond to the poses where the landmark was observed, because all other blocks will always be zero. For convenience we also store the landmark's Jacobians and residuals in the same landmark block, as shown in Figure 2 (b). This storage scheme can be applied both to the undamped and the damped Jacobian (see Section 5.3 for damping).
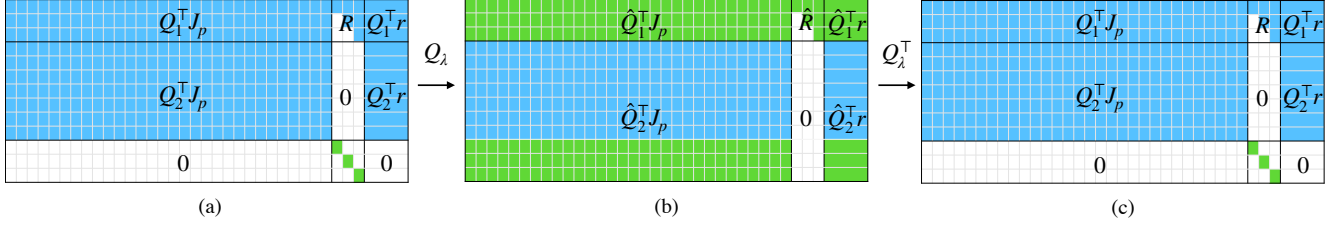
Figure 3: Illustration of the landmark damping in the Levenberg-Marquardt optimization. (a) We add three zero rows with diagonal damping for landmarks to the marginalized landmark block. (b) With 6 Givens rotations we eliminate the values on diagonal, which gives us a new landmark block with marginalized out landmark. (c) By applying the transposed rotations in reverse order and zeroing out the diagonal we can bring the landmark block to the original state. Zero entries of the landmark block are shown in white, parts that change are shown in green, and parts that stay unchanged are shown in blue.

5.2. QR decomposition

Applying a sequence of Givens rotations in-place transforms the landmark block to the marginalized state shown in Figure 2 (c). The bottom part corresponds to the reduced camera system, and the top part can be used for back substitution. This transformation can be applied to each block independently, possibly in parallel. We never have to explicitly store or compute the matrix Q; we simply apply the sequence of Givens rotations to the landmark block one by one, as they are computed. Note that alternatively we can use three Householder reflections per landmark block, with which we noticed a minor improvement in runtime.

5.3. Levenberg-Marquardt damping

The augmentation of the Jacobians by diagonal matrices as used in (6) consists of two parts that we treat differently to optimally exploit the nature of the BA problem in our implementation.

Landmark damping We first look at damping the landmark variables: rather than actually attaching a large diagonal matrix $\sqrt{\lambda}D_l$ to the full landmark Jacobian J_l, we can again work on the landmark block from Figure 2 (b) and only attach a 3×3 sub-block there, see Figure 3 (a) and (b). To simplify the expressions in figures, we slightly abuse notation when considering a single landmark and denote the corresponding parts of J_p, J_l and r in the landmark block by the same symbols. The matrices involved in the QR decomposition of the undamped system are Q_1, Q_2, R_1 and those for the damped system are marked with a hat. Note that Q and $\hat{Q}$ are closely related; the additional three diagonal entries in the damped formulation can be zeroed using only six Givens rotations, such that

$$\hat{Q} = \begin{pmatrix} \hat{Q}_1 & \hat{Q}_2 \end{pmatrix} = \begin{pmatrix} Q_1 & Q_2 & 0 \\ 0 & 0 & \mathbf{I}_3 \end{pmatrix} Q_\lambda \,, \qquad (26)$$

where Q_λ is a product of six Givens rotations. Thus, applying and removing landmark damping is computationally

cheap: we apply the Givens rotations one by one and store them individually (rather than their product Q_λ) to undo the damping later. Figure 3 illustrates how this can be done in-place on the already marginalized landmark block. This can speed up LM's backtracking, where a rejected increment is recomputed with the same linearization, but with increased damping. By contrast, for an explicit SC solver, changing the damping would mean recomputing the Schur complement from scratch.

Pose damping Marginalizing landmarks using Givens rotations in the damped formulation of (6) does not affect the rows containing pose damping. Thus, it is still the original diagonal matrix $\sqrt{\lambda}D_p$ that we append to the bottom of $\hat{Q}^\top J_p$:

$$\hat{J}_p = \begin{pmatrix} \hat{Q}_1^\top J_p \\ \hat{Q}_2^\top J_p \\ \sqrt{\lambda}D_p \end{pmatrix} \,. \qquad (27)$$

In practice, we do not even have to append the block, but can simply add the corresponding summand when evaluating matrix-vector multiplication for the CG iteration (28).

5.4. Conjugate gradient on normal equations

To solve for the optimal Δx_p^* in small and medium systems, we could use dense or sparse QR decomposition of the stacked $Q_2^\top J_p$ from landmark blocks to minimize the linear least squares objective $\|Q_2^\top J_p \Delta x_p + Q_2^\top r\|^2$. However, for large systems this approach is not feasible due to the high computational cost. Instead, we use CG on the normal equations as proposed in [2]. Other iterative solvers like LSQR [29] or LSMR [14] that can be more numerically stable than CG turned out to not improve stability for the case of bundle adjustment [8].

CG accesses the normal equation matrix $\tilde{H}_{pp}$ only by multiplication with a vector v, which we can write as

$$\tilde{H}_{pp}\, v = (\hat{Q}_2^\top J_p)^\top (\hat{Q}_2^\top J_p\, v) + \lambda D_p^2\, v \,. \qquad (28)$$

This multiplication can be efficiently implemented and well parallelized using our array of landmark blocks. Thus, we do not need to explicitly form the normal equations for the reduced least squares problem.

Still, the CG part of our solver has the numerical properties of the normal equations (squared condition number compared to the marginal Jacobian $\hat{Q}_2^\top J_p$). To avoid numeric issues when using single-precision floating-point numbers, we scale the columns of the full Jacobian to have unit norm and use a block-Jacobi preconditioner, both standard procedures when solving BA problems and both also used in the other evaluated solvers. We also note that with the Levenberg-Marquardt algorithm, we solve a strictly positive definite damped system, which additionally improves the numeric stability of the optimization.

Storing the information used in CG in square root form allows us to make sure that $\tilde{H}_{pp}$ is always strictly positive definite. As we show with our experiments (see Section 6.4), for many sequences small round-off errors during SC (explicit or implicit) render $\tilde{H}_{pp}$ to be numerically indefinite with single-precision floating-point computations.

With the computed Δx_p^* we can do back substitution for each individual landmark block independently and in parallel. We already have all the necessary information ($\hat{Q}_1^\top J_p$, $\hat{R}$, $\hat{Q}_1^\top r$) stored in the landmark blocks after marginalization.

5.5. Parallel implementation

As pointed out above, the linearization, marginalization, and back substitution can be computed independently for each landmark block. There is no information shared between landmark blocks, so we can use a simple *parallel for* loop to evenly distribute the workload between all available CPU cores. The matrix-vector multiplications that constitute the most computationally expensive part of CG can also be efficiently parallelized. In this case, multiplication results of individual landmark blocks have to be summed, so we employ the common *parallel reduce* pattern. How effective these simple parallelization schemes are is underlined by our evaluation, which shows excellent runtime performance of the square root bundle adjustment implementation, compared to both our custom and Ceres' SC solvers.

6. Experimental evaluation

6.1. Algorithms and setup

We implement our $\sqrt{BA}$ solver in C++ in single ($\sqrt{BA}$-32) and double ($\sqrt{BA}$-64) floating-point precision and compare it to the methods proposed in [2] as implemented in Ceres Solver [1]. This solver library is popular in the computer vision and robotics community, since it is mature, performance-tuned, and offers many linear solver variations. That makes it a relevant and challenging baseline

	$\sqrt{BA}$-32 (ours)	$\sqrt{BA}$-64 (ours)	explicit-32	explicit-64	ceres-implicit	ceres-explicit
solver implementation	custom				Ceres	
float precision	s	d	s	d	d	
landmark marginalization	NM		SC		SC	
RCS storage	LMB		H		–	H

Table 1: The evaluated solvers—proposed and baseline—are implemented either completely in our custom code base, or using Ceres, with single (s) or double (d) floating-point precision, using Nullspace (NM) or Schur complement (SC)-based marginalization of landmarks, and storing the reduced camera system sparsely in landmark blocks (LMB), sparsely as a reduced Hessian (H), or not at all (–).

to benchmark our implementation against. While Ceres is a general-purpose solver, it is very good at exploiting the specific problem structure as it was built with BAL problems in mind. Our main competing algorithms are Ceres' sparse Schur complement solvers, which solve the RCS iteratively by either explicitly saving $\tilde{H}_{pp}$ in memory as a block-sparse matrix (*ceres-explicit*), or otherwise computing it on the fly during the iterations (*ceres-implicit*). In both cases, the same block-diagonal of $\tilde{H}_{pp}$ that we use in $\sqrt{BA}$ is used as preconditioner. As the bottleneck is not computing Jacobians, but marginalizing points and the CG iterations, we use analytic Jacobians for our custom solvers and Ceres' exact and efficient autodiff with dual numbers. For an even more direct comparison, we additionally implement the sparse iterative explicit Schur complement solver without Ceres, sharing much of the code with our $\sqrt{BA}$ implementation. While Ceres always uses double precision, we use our custom implementation to evaluate numerical stability by considering single (*explicit-32*) and double (*explicit-64*) precision. Table 1 summarizes the evaluated configurations.

For Ceres we use default options, unless otherwise specified. This includes the scaling of Jacobian columns to avoid numerical issues [2]. Just like in our custom implementation, we configure the number of threads to be equal to the number of (virtual) CPU cores. Our Levenberg-Marquardt loop is in line with Ceres: starting with initial value 10^{-4}, we update the damping factor λ according to the ratio of actual and expected cost reduction, and run it for at most 50 iterations, terminating early if a relative function tolerance of 10^{-6} is reached. In the inner CG loop we use the same forcing sequence as Ceres, with a maximum of 500 iterations and no minimum. We run experiments on an Ubuntu
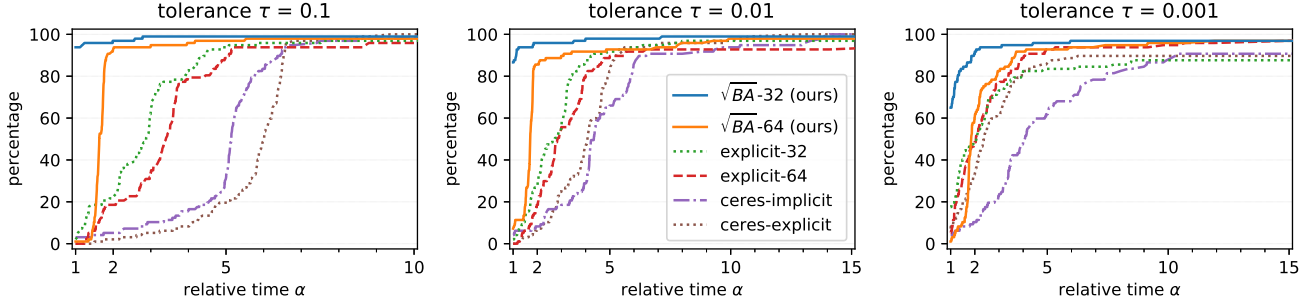
Figure 4: Performance profiles for all BAL datasets show percentage of problems solved to a given accuracy tolerance τ with increasing relative runtime α. Our proposed $\sqrt{BA}$ solver outperforms other methods across all accuracy tolerances. In single precision, the solver is about twice as fast as with double, but does not lose in accuracy, underpinning the favorable numerical properties of the square root formulation. In contrast, while the SC solver in double precision is equally accurate, this is not the case for the single-precision variant. Keep in mind that Ceres is not as tailored to the exact problem setup as our custom implementation, possibly explaining its slower performance. Note that the performance profiles are cut off at the right side of the plots to show only the most relevant parts.

18.04 desktop with 64GB RAM and an Intel Xeon W-2133 with 12 virtual cores at 3.60GHz. In our own solver implementation we rely on Eigen [16] for dense linear algebra and TBB[19] for simple *parallel for* and *parallel reduce* constructs.

6.2. Datasets

For our extensive evaluation we use all 97 bundle adjustment problems from the BAL [2] project page. These constitute initialized bundle adjustment problems and come in different groups: the *trafalgar*, *dubrovnik*, and *venice* problems originate from successive iterations in a skeletal SfM reconstruction of internet image collections [30]. They are combined with additional leaf images, which results in the thus denser *final* problems. The *ladybug* sequences are reconstructions from a moving camera, but despite this we always model all camera intrinsics as independent, using the suggested *Snavely* projection model with one focal length and two distortion parameters. Figure 5 visualizes some exemplar problems after they have been optimized.

As is common, we apply simple gauge normalization as preprocessing: we center the point cloud of landmarks at the coordinate system origin and rescale to median absolute deviation of 100. Initial landmark and camera positions are then perturbed with small Gaussian noise. To avoid close-to-invalid state, we remove all observations with a small or negative z value in the camera frame, completely removing landmarks with less than two remaining observations. We additionally employ the Huber norm with a parameter of 1 pixel for residuals (implemented with IRLS as in Ceres). This preprocessing essentially follows Ceres' BAL examples. It is deterministic and identical for all solvers by using a fixed random seed and being computed on state in double precision, regardless of solver configuration.

6.3. Performance profiles

When evaluating a solver, we are primarily interested in accurate optimization results. Since we do not have independent ground truth of correct camera positions, intrinsics, and landmark locations, we use the bundle adjustment cost as a proxy for accuracy. Lower cost in general corresponds to better solutions. But depending on the application, we may desire in particular low runtime, which can be a trade-off with accuracy. The difficulty lies in judging the performance across several orders of magnitudes in problem sizes, cost values, and runtimes (for the BAL datasets the number of cameras n_p ranges from 16 to 13682). As proposed in prior work [23, 12], we therefore use performance profiles to evaluate both accuracy and runtime jointly. The performance profile of a given solver maps the relative runtime α (relative to the fastest solver for each problem and accuracy) to the percentage of problems solved to accuracy τ. The curve is monotonically increasing, starting on the left with the percentage of problems for which the solver is the fastest, and ending on the right with the percentage of problems on which it achieves the accuracy τ at all. The curve that is more to the left indicates better runtime and the curve that is more to the top indicates higher accuracy. A precise definition of performance profiles is found in the appendix.

6.4. Analysis

Figure 4 shows the performance profiles with all BAL datasets for a range of tolerances $\tau \in \{10^{-1}, 10^{-2}, 10^{-3}\}$. We can see that our proposed square root bundle adjustment solver $\sqrt{BA}$-*64* is very competitive, yielding better accuracy than some SC-based iterative solvers, and often at a lower runtime. $\sqrt{BA}$-*32* is around twice as fast and equally accurate, which highlights the good numerical stability of the square root formulation. It clearly outperforms all other
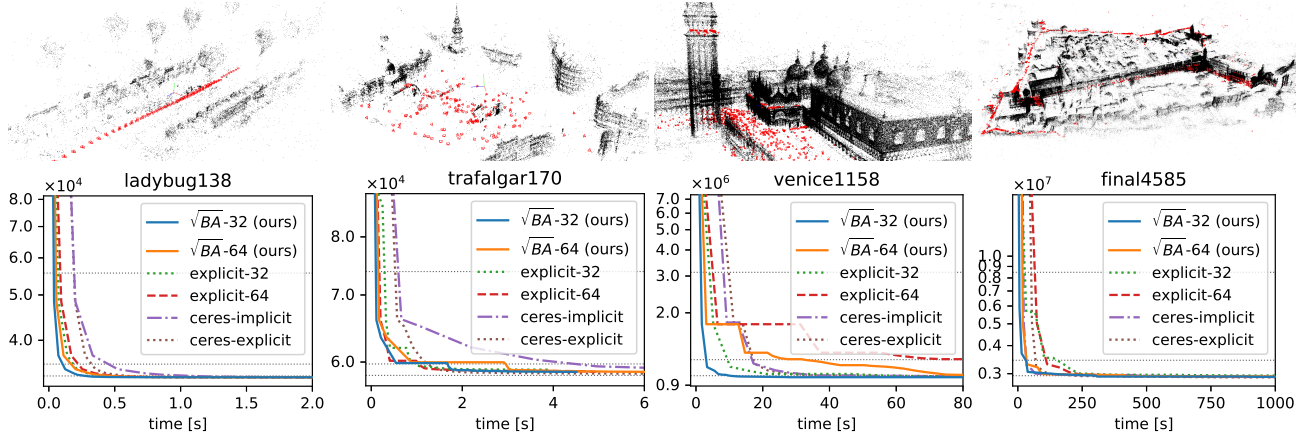
Figure 5: Convergence plots from small to large problems and rendered optimized landmark point clouds. The y-axes show the total BA cost (log scale), and the horizontal lines indicate cost thresholds for the tolerances $\tau \in \{10^{-1}, 10^{-2}, 10^{-3}\}$.
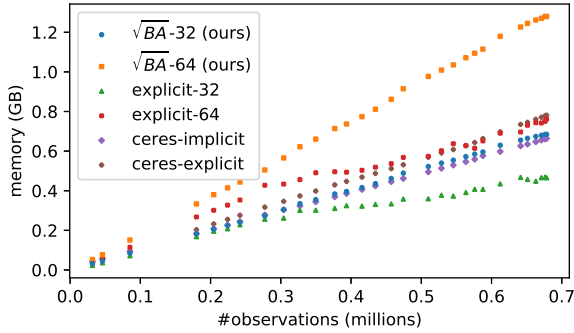

Figure 6: Memory consumption for the relatively sparse *ladybug* problems grows linearly with the problem size. The number of cameras here ranges from 49 to 1723.

solvers across all tolerances. The fact that *explicit-32* does not always reach the same accuracy as its double precision counterpart *explicit-64* indicates that SC-based solvers do not exhibit the same numerical stability. We also do not observe the same twofold speedup, which is related to the fact that *explicit-32* does have to backtrack in the LM loop significantly more often to increase damping when the Schur complement matrix becomes indefinite due to round-off errors. This happens at least once for 84 out of the 97 problems with *explicit-32* and even with *explicit-64* for 7 of the problems. With $\sqrt{BA}$, we have never encountered this.

A similar conclusion can be drawn from the convergence plots in Figure 5, which show a range of differently sized problems. For the small *ladybug* as well as the medium and large *skeletal* problems, our solver is faster. Even on the large and much more dense *final4585*, the square root solver is competitive. In the square root formulation memory and thus to some degree also runtime grows larger for denser problems—in the sense of number of observa-

tions per landmark—since a specific landmark block grows quadratically in size with the number of its observations. This is in contrast to density in the sense of number of cameras co-observing at least one common landmark, as for the SC. Still, across all BAL datasets, only for the largest problem, *final13682*, where the landmarks have up to 1748 observations, does $\sqrt{BA}$-32 run out of memory. For sparse problems, such as *ladybug*, one can see in Figure 6 that the memory grows linearly with the number of landmarks, and for $\sqrt{BA}$-32 is similar to Ceres' iterative SC solvers. In summary, while for very small problems we expect direct solvers to be faster than any of the iterative solvers, and for very large and dense problems implicit SC solvers scale better due to their memory efficiency [2], the proposed $\sqrt{BA}$ solver outperforms alternatives for medium to large problems, i.e., the majority of the BAL dataset.

7. Conclusion

We present an alternative way to solve large-scale bundle adjustment that marginalizes landmarks without having to compute any blocks of the Hessian matrix. Our square root approach $\sqrt{BA}$ displays several advantages over the standard Schur complement, in terms of speed, accuracy, and numerical stability. We have combined a very general theoretical derivation of nullspace marginalization with a tailored implementation that maximally exploits the specific structure of BA problems. Experiments comparing our solver to both a custom SC implementation and the state-of-the-art Ceres library show how $\sqrt{BA}$ can handle single-precision floating-point operations much better than the Schur complement methods, outperforming all evaluated competing approaches. We see great potential in $\sqrt{BA}$ to benefit other applications that play up to its strong performance on sparse problems, for example incremental SLAM.

References

[1] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. http://ceres-solver.org.

[2] Sameer Agarwal, Noah Snavely, Steven M Seitz, and Richard Szeliski. Bundle adjustment in the large. In *European Conference on Computer Vision (ECCV)*, pages 29–42. Springer, 2010.

[3] David S Bayard and Paul B Brugarolas. An estimation algorithm for vision-based exploration of small bodies in space. In *American Control Conference (ACC)*, pages 4589–4595. IEEE, 2005.

[4] Gerald J Bierman. *Factorization methods for discrete sequential estimation*. Courier Corporation, 2006.

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

[6] Åke Björck and Tommy Elfving. Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations. *BIT Numerical Mathematics*, 19(2):145–163, 1979.

[7] Duane C Brown. A solution to the general problem of multiple station analytical stereotriangulation. RCA-MTP data reduction technical report no. 43 (or AFMTC TR 58-8), Patrick Airforce Base, Florida, 1958.

[8] Martin Byröd and Kalle Åström. Conjugate gradient bundle adjustment. In *European Conference on Computer Vision (ECCV)*, pages 114–127, 2010.

[9] Luca Carlone, Pablo Fernandez Alcantarilla, Han-Pang Chiu, Zsolt Kira, and Frank Dellaert. Mining structure fragments for smart bundle adjustment. In *British Machine Vision Conference (BMVC)*, 2014.

[10] Luca Carlone, Zsolt Kira, Chris Beall, Vadim Indelman, and Frank Dellaert. Eliminating conditionally independent sets in factor graphs: A unifying perspective based on smart factors. In *International Conference on Robotics and Automation (ICRA)*, pages 4290–4297. IEEE, 2014.

[11] Frank Dellaert and Michael Kaess. Square Root SAM: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research (IJRR)*, 25(12):1181–1203, 2006.

[12] Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002.

[13] Chris Engels, Henrik Stewénius, and David Nistér. Bundle adjustment rules. *Photogrammetric computer vision*, 2(32), 2006.

[14] David Chin-Lung Fong and Michael Saunders. LSMR: An iterative algorithm for sparse least-squares problems. *SIAM Journal on Scientific Computing*, 33(5):2950–2971, 2011.

[15] Gene H. Golub and Charles F. van Loan. *Matrix Computations*. JHU Press, fourth edition, 2013.

[16] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[17] Magnus R Hestenes, Eduard Stiefel, et al. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49(6):409–436, 1952.

[18] J. H. Hong, C. Zach, and A. Fitzgibbon. Revisiting the variable projection method for separable nonlinear least squares problems. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5939–5947, 2017.

[19] Intel. Threading building blocks. http://threadingbuildingblocks.org, 2006.

[20] Yekeun Jeong, David Nister, Drew Steedly, Richard Szeliski, and In-So Kweon. Pushing the envelope of modern methods for bundle adjustment. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(8):1605–1617, 2011.

[21] Kurt Konolige and Willow Garage. Sparse sparse bundle adjustment. In *British Machine Vision Conference (BMVC)*, volume 10, pages 102–1, 2010.

[22] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o: A general framework for graph optimization. In *International Conference on Robotics and Automation (ICRA)*, pages 3607–3613. IEEE, 2011.

[23] Avanish Kushal and Sameer Agarwal. Visibility based preconditioning for bundle adjustment. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1442–1449. IEEE, 2012.

[24] Manolis IA Lourakis and Antonis A Argyros. SBA: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software (TOMS)*, 36(1):1–30, 2009.

[25] Peter S Maybeck. *Stochastic models, estimation, and control*. Academic press, 1982.

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

[27] Karthikeyan Natesan Ramamurthy, Chung-Ching Lin, Aleksandr Aravkin, Sharath Pankanti, and Raphael Viguier. Distributed bundle adjustment. In *International Conference on Computer Vision Workshop (ICCVW)*, pages 2146–2154, 2017.

[28] T. Okatani, T. Yoshida, and K. Deguchi. Efficient algorithm for low-rank matrix factorization with missing components and performance comparison of latest algorithms. In *International Conference on Computer Vision (ICCV)*, pages 842–849, 2011.

[29] Christopher C Paige and Michael A Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software (TOMS)*, 8(1):43–71, 1982.

[30] Jan Svoboda, Thomas Cashman, and Andrew Fitzgibbon. QRkit: Sparse, composable QR decompositions for efficient and stable solutions to problems in computer vision. In *Winter Conference on Applications of Computer Vision (WACV)*, pages 1263–1272. IEEE, 2018.

[31] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.

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

[33] Kejian Wu, Ahmed Ahmed, Georgios A Georgiou, and Stergios I Roumeliotis. A square root inverse filter for efficient vision-aided inertial navigation on mobile devices. In *Robotics: Science and Systems (RSS)*, volume 2, 2015.

[34] Yulin Yang, James Maley, and Guoquan Huang. Null-space-based marginalization: Analysis and algorithm. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 6749–6755. IEEE, 2017.

[35] Christopher Zach. Robust bundle adjustment revisited. In *European Conference on Computer Vision (ECCV)*, pages 772–787. Springer, 2014.

[36] Lei Zhou, Zixin Luo, Mingmin Zhen, Tianwei Shen, Shiwei Li, Zhuofei Huang, Tian Fang, and Long Quan. Stochastic bundle adjustment for efficient and scalable 3d reconstruction. In *European Conference on Computer Vision (ECCV)*, 2020.