Square Root Marginalization for Sliding-Window Bundle Adjustment

Nikolaus Demmel   David Schubert   Christiane Sommer   Daniel Cremers   Vladyslav Usenko
Technical University of Munich

{nikolaus.demmel,d.schubert,c.sommer,cremers,vlad.usenko}@tum.de

Abstract

In this paper we propose a novel square root sliding-window bundle adjustment suitable for real-time odometry applications. The square root formulation pervades three major aspects of our optimization-based sliding-window estimator: for bundle adjustment we eliminate landmark variables with nullspace projection; to store the marginalization prior we employ a matrix square root of the Hessian; and when marginalizing old poses we avoid forming normal equations and update the square root prior directly with a specialized QR decomposition. We show that the proposed square root marginalization is algebraically equivalent to the conventional use of Schur complement (SC) on the Hessian. Moreover, it elegantly deals with rank-deficient Jacobians producing a prior equivalent to SC with Moore-Penrose inverse. Our evaluation of visual and visual-inertial odometry on real-world datasets demonstrates that the proposed estimator is 36% faster than the baseline. It furthermore shows that in single precision, conventional Hessian-based marginalization leads to numeric failures and reduced accuracy. We analyse numeric properties of the marginalization prior to explain why our square root form does not suffer from the same effect and therefore entails superior performance.

1. Introduction

Visual odometry has been a key component in environment mapping, robot navigation and autonomous systems for a long time. With low-cost devices, such as smartphones or robot vacuum cleaners becoming increasingly prevalent in our daily lives, we see a growing need to solve odometry problems in a fast and robust manner. In addition, scalable solutions on specialized hardware require algorithms to run with limited floating point precision.
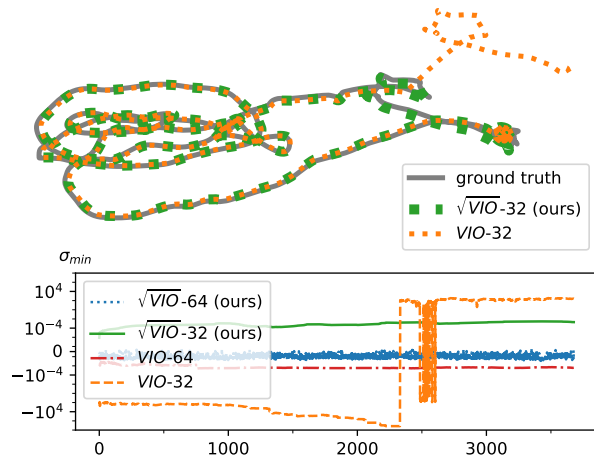
To keep the system size bounded with a fixed number

Figure 1. *Top:* estimated visual-inertial odometry trajectories on the *eurocMH01* sequence. The conventional baseline *VIO*-64 works well with double-precision floats, but fails in single precision (*VIO*-32). In contrast, the proposed square root estimator $\sqrt{VIO}$-32 in single precision retains the same accuracy. It processes the whole sequence in $26\,\mathrm{s}$ on a desktop CPU (7.1x faster than real time). *Bottom:* smallest eigenvalue $\sigma_{\min}$ of the marginalization prior Hessian H_{m} evolving over time (linear y-axis for $|\sigma_{\min}| < 10^{-8}$, logarithmic elsewhere). We expect values close to zero (positive semi-definite Hessian with gauge freedom). While the conventional (squared) formulation in single precision leads to negative eigenvalues with large magnitude, accumulating error, and (ultimately) numerical failure, the proposed square root approach has $\sigma_{\min}$ of bounded magnitude and remains stable.

of state variables over time, marginalization is a commonly used technique, where the remaining sub-problem can be interpreted in terms of a marginal distribution with the same solution as before. However, for the implementation of the marginalization prior as well as the solution of the associated optimization problem, there are multiple options. The Schur complement technique is an easy-to-implement choice that many state-of-the-art odometry and SLAM systems employ, but it relies on the Hessian matrix of the linearized system. While this is not a problem in many applications, the fact that the square of the Jacobian and thus a squared condition number are involved in SC may lead

to numerical instabilities. In Kalman filter literature, this has often been addressed by using a square root filter approach [29]. Moreover, it has recently been shown that even in an optimization-based bundle adjustment setting, matrix square roots can be exploited to increase numerical stability [6]. Following up on these findings, we propose an optimization-based formulation of visual (inertial) odometry that uses matrix square roots in the optimization stage as well as to store and update the marginalization prior. We dub the two flavors of our method $\sqrt{VO}$ and $\sqrt{VIO}$ for purely visual and visual-inertial odometry, respectively. The contributions of our paper are as follows:

- We propose a novel square root formulation for optimization-based sliding-window estimators.
- One of the components of this square root formulation is a specialized QR decomposition and we prove the analytical equivalence with marginalization based on SC and Cholesky decomposition.
- Our QR marginalization naturally includes the case of rank-deficient Jacobians.
- The proposed square root formulation enables optimization-based VO and VIO with single-precision floating point computations without loss of accuracy.
- On several real-world datasets, we systematically analyze the effect of using nullspace projection combined with our specialized QR decomposition on runtime and accuracy. The proposed square root estimator is 23% faster than the baseline and 36% faster when additionally switching to single precision.
- We release our implementation as open source: https://go.vision.in.tum.de/rootvo.

2. Related work

In what follows, we review relevant literature in visual and visual-inertial odometry, discuss filter-based and optimization-based techniques, and highlight works dealing with rank-deficient Jacobians.

Filter-based state estimation for VIO More than 60 years ago, Kalman proposed a filter [12] that set the basis for state estimation in many different applications, visual-inertial odometry being one of them. Many variations of the Kalman filter have been developed, introducing different improvements to the original, where one prominent example for VIO is the MSCKF [19]. Square root filters are able to improve numerical stability and let the system run on single-precision hardware [16, 1, 5, 29] and in particular information filters are closely related to optimization-based approaches [26]. As such, the proposed approach shares many of these advantages with the square root inverse variant of the MSCKF [29], but ours can relinearize residuals until they leave the sliding window and accommodate rank-deficient marginalization priors. Yang et al. were the first to prove the equivalence of Schur complement and nullspace marginalization (a technique using matrix square roots) from a Kalman filter perspective [30].

Optimization-based visual (inertial) odometry systems Optimization-based approaches have become the state-of-the-art in visual and visual-inertial odometry. Usually, they are implemented as a fixed-lag smoother, where old or redundant state variables are removed from the optimization window in order to keep the system real-time capable [25, 13, 7, 22, 28]. All mentioned approaches rely on marginalization using the Schur complement, which means the marginalization energy is stored as a quadratic function of the remaining variables. Recent work, however, has shown that it can be beneficial to perform marginalization in square root form [6], both numerically and in terms of runtime. While [6] only shows this for batch optimization with temporary landmark marginalization, we introduce an efficient way to permanently marginalize frame variables.

Rank-deficient Jacobians Most works on bundle adjustment and odometry systems tacitly assume full-rank Jacobians and information matrices, but this may not always be the case. Generalized inverses, generalized Schur complements and rank-deficient information matrices have been analyzed for years outside the BA or SLAM context [23, 15, 2, 4, 20, 21, 14]. Only few works propose ways to deal with rank-deficiency in SLAM: Mazuran et al. [18, 17] introduce an orthogonal projection of all involved matrices that removes the Jacobian degeneracy and Leutenegger et al. [13] use the pseudo-inverse for landmark marginalization in case the Hessian is singular.

3. Sliding-window bundle adjustment

3.1. Problem statement

3.1.1 Batch optimization problem

The goal of odometry is to estimate the system state from a set of visual, inertial, or other measurements. The state x can consist of poses, velocities, biases at different moments of time (all referred to as *frame variables* hereafter), and landmark positions. In this paper we consider visual odometry (VO) and visual-inertial odometry (VIO), but the theory described here is general and can be applied to other sensors (GPS, LIDAR). If visual measurements are involved, landmark positions are part of the optimization problem, and eventually a bundle adjustment-type problem is solved.

To estimate the optimal state x we can solve the non-linear least squares optimization problem

$$E(x) = \tfrac{1}{2}\|W^{1/2}\,\hat{r}(x)\|^2 = \tfrac{1}{2}\|r(x)\|^2 \qquad (1)$$

where $\hat{r}(x)$ is a vector function of stacked residuals and W is the weighting matrix related to the measurement

uncertainty. To simplify notation we absorb W into the weighted residual vector $r(x)$. Our implementation is based on *Basalt* [28] and we follow their residual formulation for the vision and IMU terms. (1) is an incrementally evolving batch optimization problem that grows in size and includes measurements and variables up to the current time.

3.1.2 Sliding-window energy

To keep the system real-time capable, the optimization is constrained to a small window of recent states and residuals. When new states are added, old ones are removed through marginalization, and we keep the information from residuals that depend on removed variables in an additional energy term $E_m(x)$. Together with the squared vector of active residuals r_a, this forms the sliding-window energy

$$E_{sw}(x) = \tfrac{1}{2}\|r_a(x)\|^2 + E_m(x)\,. \tag{2}$$

From here on, x contains only those variables which are active in the current window. By repeating this process of marginalizing variables when new measurements and states are added, we can keep the problem size fixed. We refer to this type of marginalization, which allows for keeping the problem size small, as *permanent* marginalization, since the marginalized variables are not reintroduced to the problem. Using permanent marginalization, we describe how (2) transforms from one point in time to the next in Sec. 3.2.

3.1.3 Hessian versus square root form

Most commonly, the marginalization energy is stored as a quadratic form of the active optimization variables:

$$E_m(x) = \tfrac{1}{2}(x - x^0)^\top H_m (x - x^0) + b_m^\top (x - x^0)\,, \tag{3}$$

where x^0 is the linearization point. Note, that strictly speaking x here only denotes a subset of the variables and H_m and b_m are of according size (see Sec. 3.2 for details).

It is also possible to store this energy in square root form:

$$E_m(x) = \tfrac{1}{2}\|r_m + J_m(x - x^0)\|^2\,. \tag{4}$$

Up to an additive constant, this is the same as (3) and we can relate the two representations by

$$H_m = J_m^\top J_m\,, \qquad b_m = J_m^\top r_m\,. \tag{5}$$

Note, that for a given H_m and b_m, J_m and r_m are not unique.

Shifting the expansion point by δx away from x^0 means updating r_m by $r_m' = r_m + J_m \delta x$, while $J_m' = J_m$. The equivalent shift in (3) can be done using

$$b_m' = b_m + H_m \delta x\,, \quad H_m' = H_m\,. \tag{6}$$

3.1.4 Optimization

To obtain a state estimate, we minimize the energy in (2),

$$\hat{x} = \arg\min_x E_{sw}(x)\,, \tag{7}$$

using Levenberg-Marquardt optimization. To do this efficiently, we exploit the sparsity of the landmark Jacobian: as is usual in bundle adjustment, we use *temporary* marginalization of all landmarks to significantly reduce the problem size, then solve the reduced camera system, and finally perform back substitution to recover the landmark positions.

3.2. Marginalization

Once the energy in (2) has been optimized for the current window, and before a new frame is added, variables are chosen for marginalization. We now derive how E_m is calculated for the upcoming window of the next time step with the traditional Hessian-based approach, followed by the proposed equivalent square root formulation in Sec. 4.

3.2.1 Linearization

Using linearized residuals, we can approximate the energy in (2) in terms of a perturbation Δx from the current (optimal) state estimate x as

$$E_{lin}(\Delta x) = \tfrac{1}{2}\|r + J\Delta x\|^2\,, \tag{8}$$

$$r = \begin{pmatrix} r_a(x) \\ r_m + J_m(x - x^0) \end{pmatrix}\,, \tag{9}$$

where x^0 is the linearization point of the old marginalization prior. The residual vector r contains the active residuals r_a and old marginalization residuals shifted to the current state estimate x. The Jacobian J contains the Jacobian of r_a and the marginalization Jacobian J_m. This constitutes a system of linear equations, with corresponding normal equations

$$H\Delta x = -b\,, \tag{10}$$

where $H = J^\top J$ and $b = J^\top r$. We now define a set of variables x_μ that we want to marginalize out, i.e., we want the energy (2) in the next window to not depend on x_μ anymore. Let x_κ be the states which share residuals with states in x_μ (or have a prior), and x_u those that are not directly connected to x_μ. With that, we can rewrite H and b as

$$H = \begin{pmatrix} H_{\mu\mu} & H_{\mu\kappa} & 0 \\ H_{\kappa\mu} & H_{\kappa\kappa}^\mu + H_{\kappa\kappa}^{\bar{\mu}} & H_{\kappa u} \\ 0 & H_{u\kappa} & H_{uu} \end{pmatrix}\,, \tag{11}$$

$$b = \begin{pmatrix} b_\mu \\ b_\kappa^\mu + b_\kappa^{\bar{\mu}} \\ b_u \end{pmatrix}\,. \tag{12}$$

The Hessian block $H_{\kappa\kappa}$ is split into one part $H_{\kappa\kappa}^\mu$ containing the Jacobians of residuals that depend on μ-variables, and $H_{\kappa\kappa}^{\bar{\mu}}$ that contains all others. The same holds for b_κ.
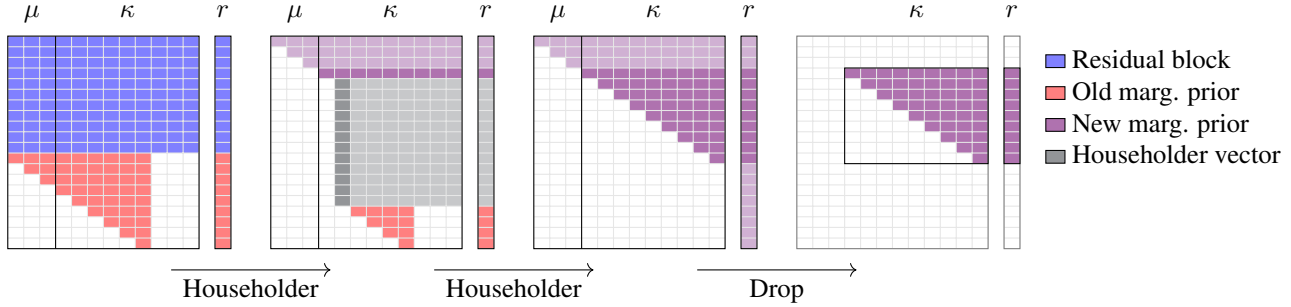
3

Figure 2. QR marginalization of frame variables. Initially, Jacobian and residual vector (after landmark marginalization) consist of residuals that depend on μ-variables and were active up to now and a marginalization prior (*left*). By successively applying Householder transformations in-place, the matrix is transformed into an upper triangular matrix (*flat*, semi-triangular in the rank-deficient case). Hereby, in each iteration, all elements except for the topmost element of the Householder vector vanish. To marginalize out the μ-variables, we drop the corresponding columns and the rows where these columns are non-zero, as well as zero rows. This results in a compact new marginalization prior (*right*). Note that the old marginalization prior on the left depicts the case where always the oldest variables are marginalized. In practice, in order to have the μ-columns on the left, variables may have to be reordered.

3.2.2 Schur complement

It can be shown by multiplication of the first line of (11) with $H_{\kappa\mu}H_{\mu\mu}^{-1}$ that solving (10) w.r.t. x_κ and x_u is equivalent to solving the reduced system

$$\begin{pmatrix} \tilde{H} + H_{\kappa\kappa}^{\bar{\mu}} & H_{\kappa u} \\ H_{u\kappa} & H_{uu} \end{pmatrix} \begin{pmatrix} \Delta x_\kappa \\ \Delta x_u \end{pmatrix} = - \begin{pmatrix} \tilde{b} + b_\kappa^{\bar{\mu}} \\ b_u \end{pmatrix}, \quad (13)$$

with

$$\tilde{H} = H_{\kappa\kappa}^{\mu} - H_{\kappa\mu}H_{\mu\mu}^{-1}H_{\mu\kappa}, \quad (14)$$

$$\tilde{b} = b_\kappa^{\mu} - H_{\kappa\mu}H_{\mu\mu}^{-1}b_\mu. \quad (15)$$

$\tilde{H}$ is called the *Schur complement* of $H_{\mu\mu}$. $\tilde{H}$ and $\tilde{b}$ only involve terms that depend on residuals containing μ-variables. Thus, they do not change if new residuals depending on κ- or u variables are added to the energy. In fact, even then, the solution of the reduced system will still be the same as if the full system including new residuals was solved.

$\tilde{H}$ and $\tilde{b}$ have been calculated using a linearization at the current state estimate x. In order to keep the system consistent, the linearization point x_κ^0 of the κ-variables may not be changed after computing $\tilde{H}$ and $\tilde{b}$ for a given x^0. These so-called first-estimates Jacobians [11] prevent the destruction of nullspaces and are commonly used [13, 7, 28]. Thus, to write the new marginalization energy as in (3), we use (6) to obtain $H_{\mathrm{m}} = \tilde{H}$ and $b_{\mathrm{m}} = \tilde{b} - \tilde{H}(x_\kappa - x_\kappa^0)$.

In practice, H_{m} and b_{m} are computed once after the optimization of the old window has converged, using only those energy terms that depend on μ-variables (always including the old prior). The result constitutes the marginalizaiton prior used in the new window. To preserve sparsity in the landmark-landmark Hessian block, we drop observations of landmarks that will stay active in frames which are about to be marginalized, before calculating $\tilde{H}$ and $\tilde{b}$. This means that landmarks are never part of the κ-variables.

4. Square root marginalization

Marginalization as presented in Sec. 3.2 is a very elegant way to keep the system size small while continuously adding new residual terms. However, the implementation using the Schur complement has some drawbacks when it comes to numerical stability, e.g., the condition number of the Hessian being squared compared to the Jacobian. In [6] it was shown that a square root formulation for temporary marginalization of variables can be beneficial for speed, accuracy and numerical stability. We now apply similar ideas to marginalization in sliding-window bundle adjustment.

4.1. Landmark marginalization

One option would be to permanently marginalize landmark and frame variables in one step. However, we know from [6] that we can use nullspace projection to exploit the special sparsity structure of the Jacobian for landmark marginalization. On the other hand, nullspace projection is not optimal for permanent marginalization of frame variables (see below). We therefore choose a two-step procedure: first, we marginalize landmarks in x_μ by projecting the Jacobians onto the nullspace Q_2 of the landmark Jacobian using QR decomposition as in [6]. In a second step we marginalize the frame variables in x_μ, taking the projected Jacobians $Q_2^\top J$ and residuals $Q_2^\top r$ as input.

4.2. Frame variable marginalization

In order to obtain a marginalization prior in square root form, we could proceed similar to [6] and QR-decompose the Jacobian J_μ. However, we want to keep the size of the marginalization prior as small as possible and we aim for a more general solution including the case where J_μ (or J) does not have full rank. To achieve this, we decompose the Jacobian $\begin{pmatrix} J_\mu & J_\kappa \end{pmatrix} = J = QR$ with a specialized

| $r = n$ | $r < n$, standard | $r < n$, flat |

☐ zero   ☒ Householder element
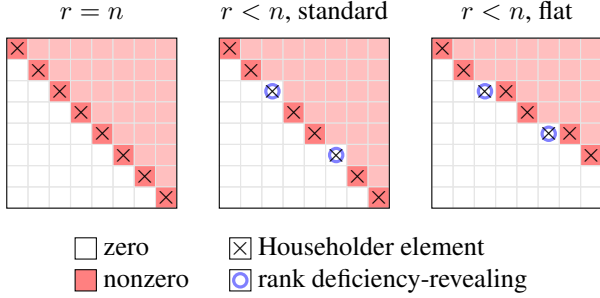■ nonzero   ⊗ rank deficiency-revealing

Figure 3. Our specialized flat QR decomposition. *Left*: for a full-rank matrix, QR decomposition results in an upper triangular matrix R with non-zero elements on the diagonal. *Middle*: if the rank r is smaller than the maximum possible rank n, standard Householder QR results in zero elements on the diagonal. *Right*: with flat QR, when a zero diagonal element occurs, the Householder element for the next column will be in the same row.

QR algorithm that uses Householder reflections and is rank-revealing without any pivoting. Fig. 2 illustrates the whole procedure in the full-rank case.

Standard Householder QR [9, p. 248] zeros the entries below the Householder element on the diagonal, column by column. This may lead to a "step" of height larger than 1 in the resulting R when that column is not linearly independent of the ones before and the Householder element remains zero (illustrated in Fig. 3, *middle*). In our specialized QR, if this occurs at matrix element j_{ik}, we proceed to the next column $k + 1$ and instead of zeroing everything below element $j_{i+1,k+1}$ like in standard QR, we keep the row index at i and zero everything below $j_{i,k+1}$ (see Fig. 3, *right*). Thus, we never get "steps" higher than one row.

The result is a valid QR decomposition with two appealing properties: when J has full rank, it is equivalent to standard Householder QR. In addition, our QR algorithm is rank-revealing without any pivoting, the rank r of J being the number of non-zero rows in R. The rank r_μ of J_μ is the number of non-zero rows in $Q^\top J_\mu$, i.e., the first n_μ columns in $Q^\top J$. Our R is in general more *flat* than that obtained by other pivoting-free QR algorithms.

For marginalization, we take R and drop the first r_μ rows, the first n_μ columns, and all zero rows at the bottom. The remaining matrix, which we call $\tilde{R}$ in the following, is a square root of the marginalization Hessian $\tilde{H}$, as will be shown in the following subsections. It has the same number n_κ of columns as $\tilde{H}$, while its number of rows may be smaller and is equal to $\mathrm{rank}(\tilde{H})$. Thus, we also fulfil the need for small prior size, and can directly read off the rank of the prior, which is not as easy for SC-based marginalization. The definition of the new marginalization energy as in (4) is now given by $J_\mathrm{m} = \tilde{R}$ and $r_\mathrm{m} = \tilde{r} - \tilde{R}(x - x^0)$, where $\tilde{r}$ contains those elements of $Q^\top r$ whose according rows

were not dropped in R and x and x^0 already have marginalized variables removed.

4.3. Equivalence to SC-based marginalization

We will only show the equivalence of our flat QR decomposition for permanent marginalization to that of Schur complement, since a proof for the equivalence of nullspace marginalization for temporary marginalization can be found in [6]. We concentrate on the equivalence of $\tilde{R}, \tilde{r}$ in square root form and $\tilde{H}, \tilde{b}$ in Hessian form. The shifts on $\tilde{r}$ and $\tilde{b}$ in order to obtain r_m and b_m are then equivalent by (6).

4.3.1 Full-rank Jacobian

Since Q in $J = QR$ is an orthogonal matrix, $H = J^\top J = R^\top R$, and R is a square root of the full Hessian. We can write R as a block matrix and define an orthogonally transformed residual vector $r' = Q^\top r$ such that $b = R^\top r'$:

$$\begin{pmatrix} R_{1\mu} & R_{1\kappa} \\ 0 & R_{2\kappa} \end{pmatrix}, \quad \begin{pmatrix} r'_1 \\ r'_2 \end{pmatrix}. \tag{16}$$

When J is a full-rank matrix, $R_{1\mu}$ and $R_{2\kappa}$ are upper triangular matrices of sizes $n_\mu \times n_\mu$ and $(N - n_\mu) \times n_\kappa$ with a number N of residuals (after landmark nullspace marginalization). The upper triangular property implies that the bottom $N - n_\kappa$ rows of $R_{2\kappa}$ are all zero, so

$$R_{2\kappa} = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}. \tag{17}$$

Note, that a matrix R with the same properties (but in general not equal) can also be obtained by LLT or LDLT decomposition of the Hessian sub-blocks related to $\{\mu, \kappa\}$, by setting $\tilde{R} = L^\top$ or $\tilde{R} = D^{\frac{1}{2}} L^\top$. So we can even store the marginalization prior in square root form after computing the Hessian, without performing any QR decomposition. We now show that marginalization with Schur complement is equivalent to keeping $R_{2\kappa}^\top$ and r'_2, and dropping the remaining components of R and r'. Using (16), we obtain

$$\begin{aligned} H_{\mu\mu} &= R_{1\mu}^\top R_{1\mu}, & H_{\kappa\kappa} &= R_{1\kappa}^\top R_{1\kappa} + R_{2\kappa}^\top R_{2\kappa}, \\ H_{\mu\kappa} &= R_{1\mu}^\top R_{1\kappa}, & H_{\kappa\mu} &= R_{1\kappa}^\top R_{1\mu}, \\ b_\mu &= R_{1\mu}^\top r'_1, & b_\kappa &= R_{1\kappa}^\top r'_1 + R_{2\kappa}^\top r'_2, \end{aligned} \tag{18}$$

where from now, $H_{\kappa\kappa}$ and b_κ denote $H_{\kappa\kappa}^\mu$ and b_κ^μ. Using these expressions and the assumption that $R_{1\mu}$ is invertible, (14) and (15) can be transformed as

$$\begin{aligned} \tilde{H} &= R_{1\kappa}^\top R_{1\kappa} + R_{2\kappa}^\top R_{2\kappa} \\ &\quad - R_{1\kappa}^\top R_{1\mu}(R_{1\mu}^\top R_{1\mu})^{-1} R_{1\mu}^\top R_{1\kappa} \\ &= R_{1\kappa}^\top R_{1\kappa} + R_{2\kappa}^\top R_{2\kappa} - R_{1\kappa}^\top R_{1\kappa} \\ &= R_{2\kappa}^\top R_{2\kappa} = \tilde{R}^\top \tilde{R}, \end{aligned} \tag{19}$$

5

and

$$\begin{aligned}
\tilde{b} &= R_{1\kappa}^\top r_1' + R_{2\kappa}^\top r_2' - R_{1\kappa}^\top R_{1\mu}(R_{1\mu}^\top R_{1\mu})^{-1} R_{1\mu}^\top r_1' \\
&= R_{1\kappa}^\top r_1' + R_{2\kappa}^\top r_2' - R_{1\kappa}^\top r_1' \\
&= R_{2\kappa}^\top r_2' = \tilde{R}^\top \tilde{r} .
\end{aligned} \quad (20)$$

The last equality in both equations is due to the fact that $R_{2\kappa}$ is empty in the lower part, see (17).

4.3.2 Rank-deficient Jacobian

Let us now assume that the Jacobian J of our problem, and thus the Hessian H, are rank-deficient, i.e., their rank is smaller than the total number of parameters we optimize. This can, for example, happen when the system is not properly constrained by an initial prior or when the prior becomes smaller than the numeric noise during operation.

We perform our rank-revealing, pivoting-free QR decomposition to compute an orthogonal $Q = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix}$ with $Q_1 \in \mathbb{R}^{N \times r_\mu}$, $Q_2 \in \mathbb{R}^{N \times (N - r_\mu)}$ and $r_\mu = \operatorname{rank}(J_\mu)$ such that the matrix

$$R = \begin{pmatrix} R_{1\mu} & R_{1\kappa} \\ 0 & R_{2\kappa} \end{pmatrix} = \begin{pmatrix} Q_1^\top \\ Q_2^\top \end{pmatrix} \begin{pmatrix} J_\mu & J_\kappa \end{pmatrix} \quad (21)$$

has a zero block on the lower left and $R_{1\mu} \in \mathbb{R}^{r_\mu \times n_\mu}$. Marginalization is done in the same way as for the full-rank case, i.e., we drop $R_{1\mu}$ and $R_{1\kappa}$, and only keep $R_{2\kappa}$, minus all zero rows at the bottom, ending up with a matrix $\tilde{R}$. The marginalization Hessian obtained this way, $\tilde{R}^\top \tilde{R}$, is the same as a pseudo-Schur complement, where in (14), (15), the inverse is replaced by a pseudo-inverse:

$$\tilde{H} = H_{\kappa\kappa} - H_{\kappa\mu} H_{\mu\mu}^+ H_{\mu\kappa} , \quad (22)$$

$$\tilde{b} = b_{\kappa\kappa} - H_{\kappa\mu} H_{\mu\mu}^+ b_\mu . \quad (23)$$

To show this equivalence, we first rewrite

$$\begin{aligned}
\tilde{R}^\top \tilde{R} &= R_{2\kappa}^\top R_{2\kappa} = J_\kappa^\top Q_2 Q_2^\top J_\kappa \\
&= J_\kappa^\top J_\kappa - J_\kappa^\top Q_1 Q_1^\top J_\kappa \\
&= H_{\kappa\kappa} - J_\kappa^\top Q_1 Q_1^\top J_\kappa .
\end{aligned} \quad (24)$$

On the other hand, for the pseudo-Schur complement, we can use the compact SVD decomposition $J_\mu = U_1 D_1 V_1^\top$ (i.e., D_1 is of size $r_\mu \times r_\mu$, see Appendix) to obtain

$$\begin{aligned}
H_{\kappa\kappa} - H_{\kappa\mu} H_{\mu\mu}^+ H_{\mu\kappa} &= H_{\kappa\kappa} - J_\kappa^\top J_\mu (J_\mu^\top J_\mu)^+ J_\mu^\top J_\kappa \\
&= H_{\kappa\kappa} - J_\kappa^\top (U_1 D_1 V_1^\top)(V_1 D_1^{-2} V_1^\top)(V_1 D_1 U_1^\top) J_\kappa \\
&= H_{\kappa\kappa} - J_\kappa^\top U_1 U_1^\top J_\kappa = H_{\kappa\kappa} - J_\kappa^\top Q_1 Q_1^\top J_\kappa ,
\end{aligned} \quad (25)$$

and similarly for $\tilde{b}$. The last equality is due to the fact that Q_1 and U_1 span the same r_μ-dimensional subspace of $\mathbb{R}^N$,

namely the space spanned by the columns of J_μ (see Appendix). Thus, using our flat QR decomposition, we obtain the same solution as by using pseudo-SC, but without the need for computing the pseudo-inverse.

4.3.3 Interpreting the generalized Schur complement

We adopt the commonly used generalization of the Schur complement technique that simply replaces matrix inverses by pseudo-inverses in the case where the Jacobian is not of full rank [13, 17]. In the following, we argue why this is a good idea: first, we define Δx_{tot} as the solution we get by computing $-H^+ b$, and let Δx_{red} be the solution that we obtain by using generalized $\tilde{H}$ and $\tilde{b}$ as in (22), (23) and solving the reduced system (13) with pseudo-inverse, (possibly) followed by back substitution for Δx_μ. Then we can use Lemma 2.3 from [14] to find that if

$$\operatorname{rank}(J_\mu) + \operatorname{rank}\begin{pmatrix} J_\kappa & J_u \end{pmatrix} = \operatorname{rank}(J) , \quad (26)$$

i.e., if the Jacobian sub-blocks corresponding to μ and $\{\kappa, u\}$ are not coupled by linearly dependent columns, the two solutions are the same, see Appendix for a proof.

Of all possible Δx that satisfy $H \Delta x = -b$, the solution Δx_{tot} obtained via H^+ is the one with smallest norm $\|\Delta x\|$. In practice, this is usually the preferred solution, as we don't want the system to drift in unobservable directions.

In case (26) does not hold, e.g. in the presence of absolute pose ambiguity, additional residuals may remove the coupled rank deficiency, and the pseudo-Schur complement approach yields the same solution for the reduced system as if a minimum-norm solution for the whole system including the new energy terms was calculated.

Note that while the property of a minimum-norm solution is very appealing, the computation of pseudo-inverses often uses SVD decomposition, which may be slow. Faster matrix decomposition techniques that can deal with singular matrices, e.g. LU or LDLT, may output solutions Δx with $\|\Delta x\| > \|\Delta x_{\text{tot}}\|$. Our flat QR combines the advantages of both: it produces the same minimum-norm solution as generalized SC *and* can be efficiently implemented.

5. Evaluation

We base our implementation on the open-source odometry *Basalt*, which is a highly efficient state-of-the-art marginalizing sliding-window odometry with KLT feature tracking as frontend and SC-based optimization and marginalization in the backend [28]. In a minor adaptation for slightly increased performance, we marginalize a feature whose track has been lost right away, and not only together with its host frame. The NS-projection for eliminating landmarks is inspired by [6], but since during optimization we solve a small system with only up to 7 keyframes and a few
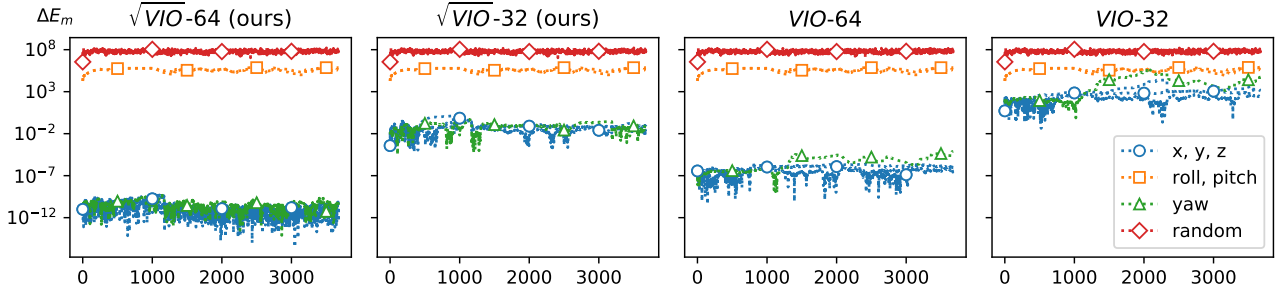
Figure 4. In visual-inertial odometry we expect 4 degrees of gauge freedom. To confirm this for the marginalized residuals, we investigate the marginalization prior cost change when perturbing the linearization point by a global translation in x, y, or z, by a global (linearized) rotation roll, pitch, or yaw, or by a random unit norm vector. While our square root marginalization leads to a consistent prior with expected nullspaces for both single and double precision, in the conventional squared form accumulating error leads to inconsistency. At around frame 2000 it appears that yaw is erroneously as observable as roll and pitch for *VIO*-32.

	$\sqrt{VIO}$-64	$\sqrt{VIO}$-32	VIO-64	VIO-32
eurocMH01	**0.093**	**0.093**	**0.093**	*0.991*
eurocMH02	**0.048**	**0.048**	**0.048**	0.048
eurocMH03	**0.051**	**0.051**	**0.051**	x
eurocMH04	**0.109**	**0.109**	**0.109**	x
eurocMH05	**0.137**	**0.137**	**0.137**	x
eurocV101	**0.043**	**0.043**	**0.043**	0.043
eurocV102	**0.048**	**0.048**	**0.048**	0.048
eurocV103	**0.058**	**0.058**	**0.058**	x
eurocV201	**0.037**	**0.037**	**0.037**	0.037
eurocV202	**0.053**	**0.053**	**0.053**	x
tumvi-corr1	**0.300**	**0.300**	**0.300**	x
tumvi-corr2	**0.426**	**0.426**	**0.426**	x
tumvi-mag1	**1.456**	*1.457*	**1.456**	x
tumvi-mag2	**0.908**	*0.920*	**0.908**	x
tumvi-room1	**0.102**	**0.102**	**0.102**	*0.104*
tumvi-room2	**0.071**	**0.071**	**0.071**	x
tumvi-slides1	**0.310**	**0.310**	**0.310**	x
tumvi-slides2	**0.759**	**0.759**	**0.759**	x

Table 1. Absolute trajectory error (in meters) for VIO shows that in contrast to the baseline, the proposed approach also works with floating point precision providing essentially the same accuracy.

	$\sqrt{VO}$-64	$\sqrt{VO}$-32	VO-64	VO-32
kitti00	**3.92**	**3.92**	**3.92**	x
kitti02	**9.72**	**9.72**	**9.72**	x
kitti03	**1.34**	**1.34**	**1.34**	1.34
kitti04	**1.22**	**1.22**	**1.22**	1.22
kitti05	**2.75**	**2.75**	**2.75**	x
kitti06	**2.61**	**2.61**	**2.61**	2.61
kitti07	*1.52*	1.53	*1.52*	1.44
kitti08	**3.85**	**3.85**	**3.85**	x
kitti09	**4.13**	**4.13**	**4.13**	x
kitti10	**1.11**	**1.11**	**1.11**	*26.12*

Table 2. Absolute trajectory error (in meters) for VO shows the same tendency as VIO in Tab. 1.

hundred landmarks, we make several adjustments that improve runtime: instead of conjugate gradient to solve the reduced camera system (RCS), we explicitly form normal equations and solve with LDLT, and we skip Jacobian scaling and landmark damping. Another alternative for solving the RCS that avoids normal equations would be QR decomposition, which we found to have higher runtime. We implement everything in one codebase, ensuring maximum comparability: the only difference between compared variants is the choice of prior storage (squared vs square root), and the algorithm for optimization and marginalization.

All variants make good use of multi-threading and use a state-of-the-art dense linear algebra library [10]. Floating point precision is indicated by suffixes, such as $\sqrt{VIO}$-64 or *VO*-32. The experiments are run on an Ubuntu 18.04

desktop with 64GB RAM and an Intel Xeon W-2133 with 12 virtual cores at 3.60GHz. VIO is evaluated on the EuRoC MAV dataset [3] and a subset of TUMVI [24]. For the KITTI odometry benchmark (training set) [8] we evaluate VO, since it does not have synchronized IMU data (*kitti01* is excluded since the optical flow of [28] fails). Additional results can be found in the Appendix.

5.1. Accuracy and runtime

We evaluate the accuracy of pose estimation with the Absolute Trajectory Error (ATE), the translational RMSE of camera positions after $\mathrm{SE}(3)$ alignment to the ground truth [27]. Tab. 1 and Tab. 2 show the ATE for VIO and VO, respectively, both for the square root and the squared implementation. It can be seen, that in double precision both variants result in the same accuracy, as does $\sqrt{VIO}$-32, while *VIO*-32 often fails numerically or results in very high ATE (same for VO). Qualitatively, the SC-based single-precision estimators perform fine initially, but usually diverge soon (see Fig. 1 *top*).

The biggest portion of total runtime is spent on optimization. There, using NS-projection is 22% faster in single

	$\sqrt{VIO}$-64	$\sqrt{VIO}$-32	VIO-64	VIO-32
eurocMH01	*23.4* / 2.5	**18.6** / 2.3	35.9 / *1.8*	33.4 / **1.7**
eurocMH02	*20.0* / 2.1	**15.6** / 1.9	31.7 / *1.5*	29.0 / **1.4**
eurocMH03	*17.6* / 1.8	**13.9** / 1.6	26.3 / **1.3**	x
eurocMH04	*13.1* / 1.3	**10.3** / 1.2	19.5 / **0.9**	x
eurocMH05	*15.0* / 1.5	**11.6** / *1.3*	22.6 / **1.1**	x
eurocV101	*15.0* / 2.2	**12.0** / *2.0*	23.6 / *1.5*	22.4 / **1.5**
eurocV102	*8.3* / 1.0	**6.8** / *0.9*	11.5 / **0.7**	10.6 / **0.7**
eurocV103	*8.3* / 1.0	**6.7** / *0.9*	11.1 / **0.7**	x
eurocV201	*12.1* / 1.4	**9.5** / *1.4*	20.8 / **1.0**	19.2 / **1.0**
eurocV202	*11.4* / 1.3	**9.3** / *1.2*	15.5 / **0.9**	x
tumvi-corr1	*24.4* / 3.2	**18.7** / *2.6*	36.7 / **2.2**	x
tumvi-corr2	*29.4* / 3.8	**22.0** / *3.1*	42.2 / **2.6**	x
tumvi-mag1	*78.1* / 10.5	**57.4** / *8.4*	112.5 / **7.0**	x
tumvi-mag2	*59.6* / 7.7	**42.2** / *6.3*	88.2 / **5.1**	x
tumvi-room1	*13.2* / 1.7	**10.0** / *1.4*	21.6 / **1.3**	19.6 / **1.3**
tumvi-room2	*12.2* / 1.8	**9.4** / *1.5*	20.2 / **1.3**	x
tumvi-slides1	*28.6* / 3.6	**20.9** / *3.0*	44.1 / **2.5**	x
tumvi-slides2	*24.8* / 3.1	**18.5** / *2.5*	38.8 / **2.1**	x

Table 3. Total runtime in seconds spent on "optimization / marginalization" in VIO. *Optimization:* with NS-projection for landmarks ($\sqrt{VIO}$-32) is almost twice as fast as the baseline using SC (VIO-64). *Marginalization:* Conventional SC is a bit faster, but this step only takes a small fraction of the overall runtime.

	proposed		ablation study					
opt.	NS+LDLT		SC+LDLT		NS+LDLT		SC+LDLT	
marg.	NS+QR		NS+QR		SC+SC		SC+SC	
precision	64	32	64	32	64	32	64	32
ATE [m]	**0.068**	**0.068**	**0.068**	**0.068**	**0.068**	0.232	**0.068**	*0.211*
real-time	6.9x	**8.2x**	5.0x	5.6x	7.1x	*7.9x*	5.2x	5.5x
t total [s]	17.9	**14.9**	24.4	21.8	17.4	*15.5*	23.7	22.2
t opt [s]	14.4	**11.4**	22.2	20.3	14.4	*11.5*	22.1	20.4
t marg [s]	1.6	1.5	1.6	*1.3*	1.4	1.4	*1.3*	**1.2**

Table 4. Different combinations of optimization and marginalization techniques, and floating-point precision for $\sqrt{VIO}$ on EuRoC. All variants store the marginalization prior in square root form (4). The shown metrics (ATE, runtime: total / optimization / marginalization) are averages over all sequences, and the real-time factor indicates how much faster the processing is compared to sequence duration. The proposed square root marginalization *NS+QR* is deciding for good accuracy in single precision, while the square root optimization *NS+LDLT* leads to best runtime.

precision compared to double, while for SC the speedup is only 8%. This is because for NS-projection we can do dense linear algebra operations on larger matrices, while an efficient SC implementation needs to exploit sparsity and operate explicitly on small matrix blocks. The larger matrix operations benefit more from SIMD instructions of modern CPUs. In total runtime, the proposed $\sqrt{VIO}$-32 is 36% faster than the baseline VIO-64 (see Tab. 3).

In an ablation study, we combine the square root prior form (4) with different optimization (*NS+LDLT* and *SC+LDLT*) and marginalization variants (*NS+QR* and *SC+SC*). Here, SC-marginalization is always immediately followed by factorizing the prior into square root form with LDLT decomposition. We can see that this factorization alone is not enough to prevent the severe degradation of accuracy in single precision. Only the combination of all proposed improvements leads to the best accuracy and runtime (see Tab. 4).

5.2. Numerical properties of marginalization prior

Analytically, H_m is positive semi-definite and has a nullspace equivalent to the non-linear system (which is one of the reasons we use first-estimates Jacobians, see Sec. 3.2.2). This nullspace contains (at least) the dimensions corresponding to the global gauge freedom of the system: global translation and yaw for VIO, and additionally roll and pitch for VO. Note, that while for optimization we add an absolute pose prior to fix the gauge, here we consider H_m without such additional prior. We analyse H_m numerically by looking at its smallest eigenvalue σ_{min} and at the

change in prior cost ΔE_m for a state perturbation ϵ around the linearization point in the direction of the expected gauge freedom, with $||\epsilon|| = 1$:

$$\Delta E_m = E_m(x^0 + \epsilon) - E_m(x^0) = \tfrac{1}{2}\epsilon^\top H_m \epsilon + \epsilon^\top b_m . \quad (27)$$

Note, that eigenvalues and cost change are always computed after converting the prior to double, and for the square root estimator we compute $H_m = J_m^\top J_m$. Fig. 1 (*bottom*) shows that with a squared formulation in single precision we get either negative eigenvalues with large magnitude (indefinite prior) or a large positive minimum eigenvalue (vanishing nullspace). Similarly, we observe in Fig. 4 that the gauge freedom appears to vanish. The proposed $\sqrt{VIO}$-32 and $\sqrt{VO}$-32 suffer from neither of these problems and are numerically stable and thus retain full accuracy.

6. Conclusion

We introduced a square root sliding-window bundle adjustment approach that is well suited for real-time visual and visual-inertial odometry applications. The method combines elimination of landmark variables using nullspace projection with a matrix square root of the Hessian for storing the maginalization prior which is in turn directly updated using a specialized QR decomposition. We proved that the specialized QR decomposition is (analytically) equivalent to Schur complement. Yet, experimental evaluation on a range of real-world datasets reveals that the proposed approach is 23% faster than the baseline. Moreover, in contrast to the baseline approach, the proposed method remains numerically stable when run in single floating point precision, leading to a combined speedup of 36% while preserving the same accuracy and robustness.

References

[1] Gerald J Bierman. *Factorization methods for discrete sequential estimation*. Courier Corporation, 2006. 2

[2] Fennell Burns, David Carlson, Emilie Haynsworth, and Thomas Markham. Generalized inverse formulas using the schur complement. *SIAM Journal on Applied Mathematics*, 26(2):254–259, 1974. 2

[3] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research (IJRR)*, 2016. 7

[4] David Carlson, Emilie Haynsworth, and Thomas Markham. A generalization of the schur complement by means of the moore–penrose inverse. *SIAM Journal on Applied Mathematics*, 26(1):169–175, 1974. 2

[5] 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. 2

[6] Nikolaus Demmel, Christiane Sommer, Daniel Cremers, and Vladyslav Usenko. Square root bundle adjustment for large-scale reconstruction. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 4, 5, 6, 12

[7] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 40(3):611–625, 2017. 2, 4

[8] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 7

[9] Gene H. Golub and Charles F. van Loan. *Matrix Computations*. JHU Press, fourth edition, 2013. 5

[10] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010. 7

[11] Guoquan P. Huang, Anastasios I. Mourikis, and Stergios I. Roumeliotis. A first-estimates jacobian EKF for improving SLAM consistency. In *Experimental Robotics*, pages 373–382. Springer Berlin Heidelberg, 2009. 4

[12] R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960. 2

[13] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research (IJRR)*, 34(3):314–334, 2015. 2, 4, 6

[14] Jianzhou Liu and Rong Huang. Generalized schur complements of matrices and compound matrices. *The Electronic Journal of Linear Algebra*, 21, 2010. 2, 6, 10

[15] George Marsaglia and George PH Styan. Rank conditions for generalized inverses of partitioned matrices. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 437–442, 1974. 2

[16] Peter S Maybeck. *Stochastic models, estimation, and control*. Academic press, 1982. 2

[17] Mladen Mazuran, Wolfram Burgard, and Gian Diego Tipaldi. Nonlinear factor recovery for long-term SLAM. *The International Journal of Robotics Research (IJRR)*, 35(1-3):50–72, 2016. 2, 6

[18] Mladen Mazuran, Gian Diego Tipaldi, Luciano Spinello, and Wolfram Burgard. Nonlinear graph sparsification for SLAM. In *Robotics: Science and Systems (RSS)*, 2014. 2

[19] 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)*, 2007. 2

[20] Diane Valerie Ouellette. Schur complements and statistics. *Linear Algebra and its Applications*, 36:187–295, 1981. 2

[21] Friedrich Pukelsheim. Information matrices in experimental design theory. 1990. 2

[22] T. Qin, P. Li, and S. Shen. VINS-Mono: A robust and versatile monocular visual-inertial state estimator. *Transactions on Robotics (T-RO)*, 34(4):1004–1020, 2018. 2

[23] Charles A Rohde. Generalized inverses of partitioned matrices. *Journal of the Society for Industrial and Applied Mathematics*, 13(4):1033–1035, 1965. 2

[24] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stueckler, and D. Cremers. The TUM VI benchmark for evaluating visual-inertial odometry. In *International Conference on Intelligent Robots and Systems (IROS)*, 2018. 7

[25] Gabe Sibley, Larry Matthies, and Gaurav Sukhatme. Sliding window filter with application to planetary landing. *Journal of Field Robotics*, 27(5):587–608, 2010. 2

[26] Hauke Strasdat, José MM Montiel, and Andrew J Davison. Visual SLAM: why filter? *Image and Vision Computing*, 30(2):65–77, 2012. 2

[27] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *International Conference on Intelligent Robot Systems (IROS)*, 2012. 7

[28] Vladyslav Usenko, Nikolaus Demmel, David Schubert, Jörg Stückler, and Daniel Cremers. Visual-inertial mapping with non-linear factor recovery. *Robotics and Automation Letters (RA-L)*, 5(2):422–429, 2019. 2, 3, 4, 6, 7

[29] 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)*, 2015. 2

[30] Yulin Yang, James Maley, and Guoquan Huang. Null-space-based marginalization: Analysis and algorithm. In *International Conference on Intelligent Robots and Systems (IROS)*, 2017. 2

A. Proofs and mathematical properties

A.1. Pseudo-Schur complement and SVD

First, we show the properties we use in Section 4.3.2 when proving that our proposed specialized QR decomposition is equivalent to using pseudo-Schur complement. This includes in particular the definition of the compact SVD for a rank-deficient matrix together with a definition of the Moore-Penrose inverse, as well as a result on subspaces of $\mathbb{R}^n$ spanned by matrix columns.

Definition 1. Let $J \in \mathbb{R}^{n \times k}$ and $\mathrm{rank}(J) = r \le k$. The *compact* singular value decomposition (SVD) of J is of the form

$$J = U_1 D_1 V_1^\top , \tag{28}$$

where $U_1 \in \mathbb{R}^{n \times r}$, $D_1 \in \mathbb{R}^{r \times r}$, and $V_1 \in \mathbb{R}^{k \times r}$. D_1 is an invertible diagonal matrix with positive entries, $U_1^\top U_1 = V_1^\top V_1 = I_r$.

Thus, by definition of the compact SVD, the columns of U_1 span the column space of J. For the compact SVD of $J^\top J$, we get $V_1 D_1^2 V_1^\top$.

Definition 2. The *Moore-Penrose inverse* (also *pseudo-inverse*) of a matrix with compact SVD $U_1 D_1 V_1^\top$ is defined as

$$(U_1 D_1 V_1^\top)^+ = V_1 D_1^{-1} U_1^\top . \tag{29}$$

Thus, the pseudo-inverse of $(J^\top J)$ is given by

$$(J^\top J)^+ = V_1 D_1^{-2} V_1^\top . \tag{30}$$

Lemma 1. Let $Q, U \in \mathbb{R}^{n \times r}$, and let the columns of Q and U span the same r-dimensional subspace of $\mathbb{R}^n$. Further, let both Q and U have mutually orthogonal columns of norm 1, i.e., $Q^\top Q = U^\top U = I_r$. Then, the following holds:

$$QQ^\top = UU^\top . \tag{31}$$

Proof. Since the columns of Q and U span the same space, each column of Q can be written as a linear combination of the columns of U and vice versa. Thus, there is a matrix M such that $Q^\top = MU^\top$ and $U^\top = M^{-1}Q^\top$. As $Q^\top Q = U^\top U = I_r$, $M = Q^\top U$ and $M^{-1} = U^\top Q = M^\top$. Thus, M is orthogonal, yielding

$$QQ^\top = UM^\top MU^\top = UU^\top . \tag{32}$$

$\square$

A.2. Equivalence of pseudo-inverse and pseudo-Schur complement

In Sec. 4.3.3, we claim that under certain conditions, solving the full system using Moore-Penrose inverse is equivalent to using the generalized Schur complement followed by solving the reduced system with Moore-Penrose inverse. Moreover, a potential backsubstitution for the μ-variables can also be achieved using a Moore-Penrose inverse instead of an inverse:

$$\Delta x_{\mu,\mathrm{red}} = H_{\mu\mu}^+ (b_\mu - H_{\mu\kappa} \Delta x_{\kappa,\mathrm{red}}) . \tag{33}$$

In the following, we will formalize and prove this statement.

Theorem 2. *Let* (26) *hold, and let* Δx_{tot} *and* Δx_{red} *be defined as in Sec.* 4.3.3. *Then,*

$$\Delta x_{tot} = \Delta x_{red} . \tag{34}$$

Proof. We start by noting that

$$\mathrm{rank}(J_\mu) = \mathrm{rank}(H_{\mu\mu}) , \tag{35}$$

$$\mathrm{rank}\begin{pmatrix} J_\kappa & J_u \end{pmatrix} = \mathrm{rank}\begin{pmatrix} H_{\kappa\kappa} & H_{\kappa u} \\ H_{u\kappa} & H_{uu} \end{pmatrix} =: r_{\kappa u} , \tag{36}$$

$$\mathrm{rank}(J) = \mathrm{rank}(H) . \tag{37}$$

Thus, we can rewrite (26) as

$$\mathrm{rank}(H) = \mathrm{rank}(H_{\mu\mu}) + r_{\kappa u} , \tag{38}$$

and apply Lemma 2.3 from [14] with $A_{11} = H_{\mu\mu}$. This Lemma gives us a block-matrix expression for the pseudo-inverse H^+ of H:

$$H^+ = \begin{pmatrix} A & -B^\top \\ -B & S^+ \end{pmatrix} , \tag{39}$$

$$S = \begin{pmatrix} \tilde{H} + H_{\kappa\kappa}^{\bar{\mu}} & H_{\kappa u} \\ H_{u\kappa} & H_{uu} \end{pmatrix} , \tag{40}$$

$$B = S^+ \begin{pmatrix} H_{\kappa\mu} \\ 0 \end{pmatrix} H_{\mu\mu}^+ , \tag{41}$$

$$A = H_{\mu\mu}^+ + H_{\mu\mu}^+ \begin{pmatrix} H_{\mu\kappa} & 0 \end{pmatrix} B . \tag{42}$$

If we now compute $-H^+ b$ and look at the κ- and u-components, we get

$$\begin{pmatrix} \Delta x_{\kappa,\mathrm{tot}} \\ \Delta x_{u,\mathrm{tot}} \end{pmatrix} = Bb_\mu - S^+ \begin{pmatrix} b_\kappa \\ b_u \end{pmatrix}$$
$$= S^+ \begin{pmatrix} H_{\kappa\mu} H_{\mu\mu}^+ b_\mu - b_\kappa \\ -b_u \end{pmatrix} = -S^+ \begin{pmatrix} \tilde{b} + b_\kappa^{\bar{\mu}} \\ b_u \end{pmatrix} , \tag{43}$$

which is exactly the solution of (13), i.e.,

$$\begin{pmatrix} \Delta x_{\kappa,\mathrm{tot}} \\ \Delta x_{u,\mathrm{tot}} \end{pmatrix} = \begin{pmatrix} \Delta x_{\kappa,\mathrm{red}} \\ \Delta x_{u,\mathrm{red}} \end{pmatrix} \tag{44}$$

Similarly, from

$$\Delta x_{\mu,\mathrm{tot}} = -Ab_\mu + B^\top \begin{pmatrix} b_\kappa \\ b_u \end{pmatrix} , \tag{45}$$

after some steps, one obtains the back substitution formula (33)

$$\Delta x_{\mu,\mathrm{tot}} = H_{\mu\mu}^+ (b_\mu - H_{\mu\kappa} \Delta x_{\kappa,\mathrm{red}}) = \Delta x_{\mu,\mathrm{red}} . \tag{46}$$

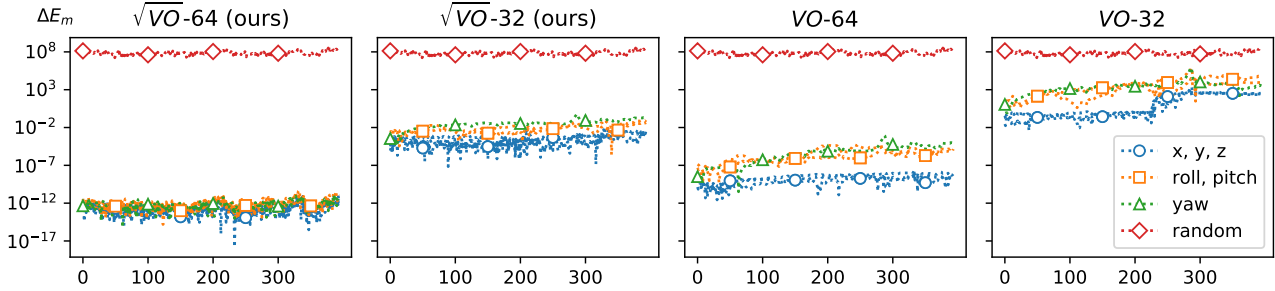(44) and (46) together conclude the proof. $\square$

Figure 5. Where for visual-inertial odometry we expect 4 degrees of gauge freedom, for stereo visual odometry roll and pitch are not observable and we expect 6 degrees of gauge freedom. The plots show the marginalization prior cost change ΔE_{m} for VO on *kitti10* when perturbing the linearization point. For that, we consider perturbations by a global translation (in x, y, or z), by a global (linearized) rotation (roll, pitch, or yaw), or by a random unit norm vector. While our square root marginalization leads to a consistent prior with expected nullspaces for both single and double precision, in the conventional squared form accumulating error leads to inconsistency. Similar to the VIO case (compare Fig. 4), here for *VO*-32 the prior over time erroneously appears to make the global pose observable, indicated by large cost change by perturbations in gauge direction. In particular, after around 200 keyframes there is a noticeable increase, which also coincides with worsened pose estimation (see Fig. 7).

	$\sqrt{VO}$-64	$\sqrt{VO}$-32	*VO*-64	*VO*-32
kitti00	*29.5 / 2.7*	**23.6 / 2.2**	50.2 / *2.3*	x
kitti02	*32.0 / 3.0*	**25.0 / 2.3**	53.2 / *2.4*	x
kitti03	*5.2 / 0.6*	**4.3 / 0.5**	9.4 / **0.5**	9.0 / **0.5**
kitti04	*1.5 / 0.2*	**1.2 / 0.1**	2.6 / **0.1**	2.5 / **0.1**
kitti05	*18.0 / 1.7*	**15.0 / 1.4**	31.1 / *1.5*	x
kitti06	*5.8 / 0.6*	**4.8 / 0.5**	9.8 / 0.6	9.3 / 0.6
kitti07	*6.3 / 0.7*	**5.3 / 0.6**	11.2 / **0.6**	10.7 / **0.6**
kitti08	*26.3 / 2.5*	**21.2 / 2.0**	44.2 / *2.1*	x
kitti09	*10.1 / 1.0*	**8.0 / 0.8**	16.7 / **0.8**	x
kitti10	*6.9 / 0.7*	**5.5 / 0.6**	11.6 / **0.6**	9.7 / **0.6**

Table 5. Total runtime in seconds spent on "optimization / marginalization" in VO. *Optimization:* NS-projection for landmarks ($\sqrt{VO}$-32) is almost twice as fast as the baseline using SC (*VO*-64). *Marginalization:* conventional SC may be slightly faster, but this step only takes a small fraction of the overall runtime.

Note on square root of the κu-system While we have shown that $\tilde{H} = \tilde{R}^\top \tilde{R}$ and $\tilde{b} = \tilde{R}^\top \tilde{r}$, to complete the square root formulation, a square root of the system including u-variables as in (13) and (40) is given by

$$\tilde{R}_{\kappa u} = \begin{pmatrix} J_\kappa^{\bar{\mu}} & J_u^{\bar{\mu}} \\ \tilde{R} & 0 \end{pmatrix}, \qquad \tilde{r}_{\kappa u} = \begin{pmatrix} r^{\bar{\mu}} \\ \tilde{r} \end{pmatrix}. \qquad (47)$$

B. Additional analysis of VO results

In this section we include additional results supporting the claims of the main paper. While these are for the same datasets, we expand upon some of the analysis that was omitted due to limited space. Specifically, we show runtimes, the ablation study, as well as the nullspace and eigenvalue analysis also for VO on the KITTI dataset. Qualita-

	proposed		ablation study					
opt.	NS+LDLT		SC+LDLT		NS+LDLT		SC+LDLT	
marg.	NS+QR		NS+QR		SC+SC		SC+SC	
precision	64	32	64	32	64	32	64	32
ATE [m]	**3.216**	**3.216**	**3.216**	**3.216**	*3.217*	3.293	**3.216**	3.479
real-time	9.4x	**9.8x**	8.0x	8.6x	9.4x	*9.6x*	8.2x	8.6x
t total [s]	24.3	*23.3*	28.7	26.5	24.1	**23.2**	28.2	26.5
t opt [s]	14.2	*11.3*	24.2	22.6	14.1	**11.0**	24.2	22.2
t marg [s]	1.4	*1.1*	1.2	**1.0**	1.4	1.3	1.2	1.2

Table 6. Different combinations of optimization and marginalization techniques, and floating-point precision for $\sqrt{VO}$ on KITTI. All variants store the marginalization prior in square root form (4). The shown metrics (ATE, runtime: total / optimization / marginalization) are averages over all sequences, and the real-time factor indicates how much faster the processing is compared to sequence duration. The proposed square root marginalization *NS+QR* is deciding for good accuracy in single precision, while the square root optimization *NS+LDLT* leads to best runtime.

tively, these are similar to the VIO results from the main paper and thus we draw the same conclusions.

Tab. 5 shows runtimes for optimization and marginalization for VO (compare VIO results in Tab. 3). It can be seen that optimization takes a much bigger portion of total runtime than marginalization, that for the proposed single-precision solver $\sqrt{VO}$-32 it is around twice as fast as the competing baseline *VO*-64, and that the square root formulation benefits more in terms of runtime from switching from double to single precision.

Tab. 6 shows the same ablation study as Tab. 4, but for VO instead of VIO. Note that for KITTI, the twofold improvement in optimization runtime is not fully reflected in an improvement of total runtime. The reason is that here the
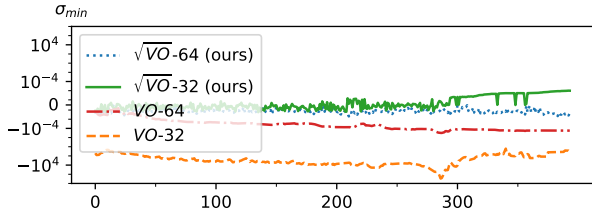
Figure 6. Smallest eigenvalue $\sigma_{\min}$ of the marginalization prior Hessian H_m evolving over time for VO on *kitti10* (linear y-axis for $|\sigma_{\min}| < 10^{-8}$, logarithmic elsewhere). We expect values close to zero (positive semi-definite Hessian with gauge freedom). While the conventional (squared) formulation in single precision leads to negative eigenvalues with large magnitude (exceeding 10^8), accumulating error, and (ultimately) numerical failure, the proposed square root approach has $\sigma_{\min}$ of bounded magnitude (less than 10^{-4}) and remains stable.
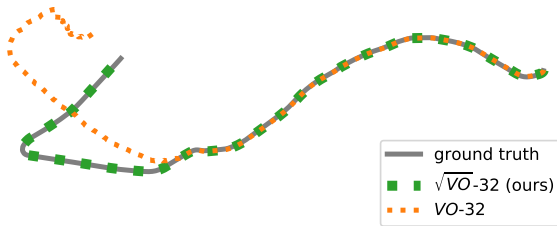

Figure 7. Estimated visual odometry trajectories on the *kitti10* sequence. The conventional baseline *VO*-64 works well with double-precision floats, but fails in single precision (*VO*-32). In contrast, the proposed square root estimator $\sqrt{VO}$-32 even in single precision retains the same accuracy.

optical flow, which is computed in a single parallel thread, becomes the bottleneck. However, the improved optimization runtime still means the required compute power is reduced. Overall, also for VO we conclude that only the combination of all proposed improvements leads to best accuracy and runtime.

The analysis of numerical properties of the marginalization prior Hessian of VO on *kitti10* reveals similar behaviour to VIO (see Sec. 5.2). For the squared formulation in single precision the marginalization prior becomes numerically indefinite (Fig. 6, compare VIO results in Fig. 1 *bottom*) and gauge freedom vanishes (Fig. 5, compare VIO results in Fig. 4). While initially the pose estimation works fine, at some point the accumulating error leads to bad state estimates and ultimately numerical failure (Fig. 7, compare Fig. 1 *top*). In contrast, the proposed $\sqrt{VO}$ has the same accuracy in both single and double precision, at a significantly reduced computational cost.

C. Notes on memory overhead

The main memory requirement of our optimization and marginalization comes from the dense landmark blocks, where we perform QR on the Jacobians in-place to marginalize landmarks. [6] reports around twice the memory use compared to SC for sparse BA problems and mentions memory to be the limiting factor for large dense problems. However, for us the number of keyframes and number of observations per landmark are bounded in the sliding window and thus memory use is not a major concern.

For example, for VIO on Euroc MH01 we have at most 4033 observations across all landmarks, and at most 7 keyframes (3 with IMU, state size 15, and 4 pose-only, state size 6), so the Jacobians have in total 8066 rows and 73 columns (3+1 extra for landmark+residual), giving an approximate upper bound of 2.4MB with 32bit floats. Measuring the actual difference in peak memory between the single and double precision variants reveals 1.3MB for the square root solver and 0.9MB for the SC solver, while the vast majority of process peak memory at around 300MB is spent in other parts of the (not memory-optimized) system (e.g. cached image queue, logging, etc...).

A memory-conscious implementation could in fact reduce the required landmark-block memory by doing a one-pass over landmarks that linearizes, marginalizes and accumulates the RCS Hessian using scratch memory. Only 3 rows per landmark for back-substitution would need to be stored.