A Practical Guide to Marginalization for Nonlinear Least Squares on Differentiable Manifolds

Evan Levine

February 3, 2023

Abstract

Many state estimation algorithms in robotics rely on nonlinear least squares optimization procedures that require permanently marginalizing out a subset of variables. This operation can be time-consuming and error-prone to implement as well as specialized to manifolds, graph structures, or problem instances. In addition, key details remain ambiguous including generalizations to manifold state spaces and numerical considerations. This report is intended to serve as a practical guide for implementers and was inspired by related work in generic sensor fusion. It presents a description of marginalization for nonlinear least squares on graphs in the framework of $\boxplus$-manifolds. In software, these abstractions allow users to marginalize out variables with the Ceres solver in a natural way and experiment with algorithmic choices. Users implement $\boxplus/\boxminus$ operators and Jacobians for each primitive manifold and marginalize in one function call. Numerical experiments demonstrate sensor fusion algorithm development and compare algorithmic choices. Code is available at github.com/evanlev/ceres-solver-ext.

1 Introduction

In many robotics applications, it is desirable to solve for a subset of variables at reduced computational cost while eliminating a set of nuisance variables. One example is bundle adjustment, which can be concerned with estimating only poses or only geometry. In sensor fusion, the sequential nature of the problem usually requires removing some variables to keep the problem size bounded. An example is visual odometry which adds and eliminates variables to maintain a sliding window of recent states [1, 2, 3, 4].

It is possible to formulate generic sensor fusion algorithms in terms of three abstractions: manifolds, nonlinear least squares, and graphs. Manifolds are mathematical sets that represent variables of interest such as pose, directions, and elements of the scene. Their smooth geometry affords a bidirectional mapping between a local neighborhood on the manifold and Euclidean space, enabling the application of sensor fusion algorithms based on local operations. The variables of interest have dependencies that are represented as a graph. Many practical real-time esimators perform nonlinear least squares optimization, which balances robustness, efficiency, and accuracy.

[5] defined a broad class of manifolds for sensor fusion called $\boxplus$-manifolds. The definition formalizes the notion of steps on the manifold in terms of a $\boxplus$ and $\boxminus$ operator. They further describe generic nonlinear least squares optimization, the extended Kalman filter, and the unscented Kalman filter in terms of $\boxplus$-manifolds. [6] extend this to the interacting multiple model filter. [7] describe Kalman filtering on differentiable manifolds. Simultaneously, optimization-based sensor

fusion has gained recent interest due to its performance and flexibility. In other work, marginalization in nonlinear least squares has been described for Euclidean space or without fully addressing the manifold structure [8, 9, 10].

Many libraries for bundle adjustment and sensor fusion have been developed. g2o [11] and Ceres [12] are popular graph optimization libraries that focus on solving bundle adjustment problems but do not provide generic marginalization. In the spirit of this report, Ceres offers a manifold API that allows separating basic manifold operations from the cost function. GTSAM [13] is a sensor fusion library that uses a factor graph formulation and also provides marginalization with some limitations on possible numerical implementations. Related work has described marginalization on the Bayes tree [14]. The SymForce library [15] uses a factor graph formulation but focuses on symbolic computation and code generation. The Manifold ToolKit [8] provides abstractions for manifolds with an optimizer and unscented Kalman filter, but it does not provide a framework for generic marginalization in nonlinear least squares problems. These libraries provide a rich language for users to manipulate problems. They implement advanced optimization techniques that exploit sparse structure, cache locality, vectorization, and various other techniques to achieve good performance for prototyping.

A practical challenge motivating this document is that marginalization is highly complex and error-prone to work out analytically, implement, and rigorously test. Generic implementations are one way to address these challenges. To meet the performance requirements for real-time systems, more specialized implementations can be required. Existing descriptions of marginalization in literature provide limited guidance to practitioners taking either approach. Most importantly, none explicitly generalize to manifold state spaces. Marginalization has multiple numerical implementations such as square root and Schur-complement-based marginalization. In Schur-complement-based marginalization, there are multiple ways of computing the required pseudoinverses including eigendecomposition and variants of the Cholesky factorization. It is useful to practitioners to have an implementation that allows exploring tradeoffs in these algorithmic choices. A common mathematical description is also useful as guidance for alternative implementations.

Our aim is to provide a detailed, explicit, and general description of marginalization with a powerful implementation that follows the math. Building on [5], we encapsulate the manifold structure using the $\boxplus$ and $\boxminus$ operators. We describe numerical implementations based on square-root and Schur-complement-based marginalization in a common framework. We elaborate on implementation details, including a novel use for modified Cholesky factorizations. We make available a public implementation that extends the Ceres library with generic marginalization, only requiring one additional manifold operation to be implemented. Experiments demonstrate its flexibility and ease of use.

2 Preliminaries

2.1 $\boxplus$-manifolds

We briefly present some preliminaries related to $\boxplus$-manifolds. For further details, see [5]. Manifolds are mathematical sets that have smooth geometry. The central concept of a $\boxplus$ manifold is the $\boxplus$ and $\boxminus$ operators, which define a way to take local steps on the manifold. For a manifold $\mathcal{M} \subset \mathbb{R}^s$ of dimension n, the operators are

$$\boxplus : \mathcal{M} \times \mathbb{R}^n \to \mathcal{M} \tag{1}$$

$$\boxminus : \mathcal{M} \times \mathcal{M} \to \mathbb{R}^n . \tag{2}$$

These operators generalize the familiar addition and subtraction operators and satisfy four properties enumerated in [5]. While $\boxplus$ is differentiable on the manifold, it need not be continuous in its left operand.

All objects are representable in software in terms of floating-point numbers. A key feature is that the implementations of $\boxplus$-manifolds can be opaque to the algorithm. From a software perspective, such a separation of concerns has advantages for implementation and testing.

2.2 Nonlinear least squares on manifolds

Consider the constrained optimization problem

$$\min_{x} \quad C(x) \tag{3}$$

$$\text{subject to} \quad x_i \in \mathcal{M}_i, i = 1, \ldots, n, \tag{4}$$

where x consists of variables $\{x_i\}_{i=1}^{\ldots,n}$ and $\mathcal{M}_i$ is a $\boxplus$-manifold.

In nonlinear least squares, the objective takes the form

$$C(x) = \frac{1}{2} \sum_i \|f_i(x_{i_1}, \ldots, x_{i_k})\|^2, \tag{5}$$

where $f_i(\cdot)$, called the residual function, depends on the parameter blocks $x_{i_1}, \ldots, x_{i_k}$. Robust loss functions and constraints can be included, and this topic is revisited in Section 4.3.

We consider nonlinear least squares solvers that express each variable x_i in local coordinates as $\breve{x}_i \boxplus \delta_i$, iteratively minimizing the following objective with respect to all δ_i simultaneously:

$$\frac{1}{2} \sum_i \|f_i(\breve{x}_{i_1} \boxplus \delta_{i_1}, \ldots, \breve{x}_{i_k} \boxplus \delta_{i_k})\|^2. \tag{6}$$

Each variable may belong to one of a small set of primitive manifolds (e.g. $SO(3), \mathbb{R}^3$) or a Cartesian product of manifolds. At a minimum, the $\boxplus$ and $\boxminus$ operators must be implemented for each primitive manifold as well as the Jacobian of $\boxplus$ with respect to δ. Note that [7] associate additional operators to $\boxplus$ for the same manifold, which can offer some convenience in implementation to the user in some cases for Kalman filtering. We base our approach on a single $\boxplus$ operator per manifold, which is not restrictive for modeling problems and follows the formulation of some widely-used nonlinear least squares solvers.

Factor graphs formalize the inter-variable dependencies expressed by Equation 6. For details, see [16].

3 Generic Marginalization

3.1 General setup

We aim to solve for a block of variables excluding the variables to be eliminated, which we denote $x_m \in \mathbb{R}^m$. In loose terms, elimination corresponds to approximating the marginal that would be defined by “integrating out" x_m in the associated density or minimizing it out in Equation 5. We can partition the variables into a block x_m, the variables related to them by residual functions (their Markov blanket), denoted by the block $x_b \in \mathbb{R}^b$, and the remaining variables $x_r \in \mathbb{R}^b$:

$$x = (x_m, x_b, x_r). \tag{7}$$

Let ∂x_m be the index set for the residual functions involving only x_m. We can write the objective as a decomposition into two

$$C(x) = C_{bm}(x_b, x_m) + C_{br}(x_b, x_r), \tag{8}$$

where

$$C_{bm}(x_b, x_m) = \frac{1}{2} \sum_{i \in \partial x_m} \|f_i(x_b, x_m)\|^2 \tag{9}$$

$$C_{br}(x_b, x_r) = \frac{1}{2} \sum_{i \notin \partial x_m} \|f_i(x_b, x_r)\|^2, \tag{10}$$

x_b, x_m, and x_r belong to a product of $\boxplus$-manifolds, which are, naturally, $\boxplus$-manifolds as proven in [5]. Since x_m lies on a $\boxplus$-manifold and the $\boxplus$ operator is surjective, we can make a change of variables from x_m to the tangent-space vector $\delta_m \in \mathbb{R}^m$ at the point $\breve{x}_m$:

$$x_m = \breve{x}_m \boxplus \delta_m \tag{11}$$

Accordingly, the objective can be rewritten

$$C(\delta_m, x_b, x_r) = c_{bm}(x_b, \delta_m) + C_{br}(x_b, x_r), \tag{12}$$

where

$$c_{bm}(x_b, \delta_m) = C_{bm}(x_b, \breve{x}_m \boxplus \delta_m) \tag{13}$$

Assume that the residual functions are differentiable at the linearization point $\breve{x}_m, \breve{x}_b$. This allows making the following linear approximation for the first term in Equation 8.

$$\begin{aligned} c_{bm}(x_b, \delta_m) &= \frac{1}{2} \sum_{i \in \partial x_m} \|f_i(x_b, \breve{x}_m \boxplus \delta_m)\|^2 && (14) \\ &\approx \frac{1}{2} \sum_{i \in \partial x_m} \|f_i(\breve{x}_b, \breve{x}_m) + J_{b,i}(x_b \boxminus \breve{x}_b) \\ &+ J_{m,i}\delta_m\|^2, && (15) \\ &\triangleq \tilde{c}_{bm}(x_b, \delta_m), && (16) \end{aligned}$$

where $\delta_m \in \mathbb{R}^m$ is an increment in the tangent space $x_m = \breve{x}_m \boxplus \delta_m$ and x_b respectively and the Jacobians are

$$\begin{aligned} J_{m,i} &\triangleq \frac{\partial f_i}{\partial x_b}(\breve{x}_b, \breve{x}_m)\frac{\partial x_b}{\partial \delta_b}(\breve{x}_b) && (17) \\ J_{b,i} &\triangleq \frac{\partial f_i}{\partial x_m}(\breve{x}_b, \breve{x}_m)\frac{\partial x_m}{\partial \delta_m}(\breve{x}_m) && (18) \\ & && (19) \end{aligned}$$

Define the stacked Jacobians and residuals

$$J_m \triangleq \begin{bmatrix} J_{m,1}^T & J_{m,2}^T & \dots & J_{m,|\partial x_m|}^T \end{bmatrix}^T \tag{20}$$

$$J_b \triangleq \begin{bmatrix} J_{b,1}^T & J_{b,2}^T & \dots & J_{b,|\partial x_m|}^T \end{bmatrix}^T \tag{21}$$

$$f \triangleq \begin{bmatrix} f_1(\breve{x}_b, \breve{x}_m)^T & \dots & f_{|\partial x_m|}(\breve{x}_b, \breve{x}_m)^T \end{bmatrix}^T. \tag{22}$$

3.2 Marginalization via the Schur complement

Next, we derive the cost after marginalization via Schur complement. $\tilde{c}_{bm}$ can be rewritten as

$$\begin{aligned}\tilde{c}_{bm}(x_b, \delta_m) =& \frac{1}{2}\begin{bmatrix} x_b \boxminus \check{x}_b \\ \delta_m \end{bmatrix}^T \begin{bmatrix} \Lambda_{bb} & \Lambda_{mb}^T \\ \Lambda_{mb} & \Lambda_{mm} \end{bmatrix} \begin{bmatrix} x_b \boxminus \check{x}_b \\ \delta_m \end{bmatrix} \\ &+ \frac{1}{2}\|f\|^2 + \begin{bmatrix} g_b \\ g_m \end{bmatrix}^T \begin{bmatrix} x_b \boxminus \check{x}_b \\ \delta_m \end{bmatrix}, \end{aligned} \tag{23}$$

where

$$g_m = J_m^T f \tag{24}$$

$$g_b = J_b^T f \tag{25}$$

$$\Lambda_{mb} = J_m^T J_b \tag{26}$$

$$\Lambda_{bb} = J_b^T J_b \tag{27}$$

$$\Lambda_{mm} = J_m^T J_m \tag{28}$$

Equation 23 is a quadratic approximation of the objective, which can be analytically minimized with respect to δ_m, yielding

$$\delta_m^* \triangleq \arg\min_{\delta_m} \tilde{c}_{bm}(x_b, \delta_m) \tag{29}$$

$$= -\Lambda_{mm}^{\dagger}(g_{mm} + \Lambda_{mb}(x_b \boxminus \check{x}_b)), \tag{30}$$

where $\Lambda_{mm}^{\dagger}$ is the pseudoinverse of Λ_{mm}.

Substituting this into $\tilde{c}_{bm}(\cdot, \cdot)$ yields

$$\begin{aligned}\tilde{c}_{bm}(x_b, \delta_m^*) =& \frac{1}{2}(x_b \boxminus \check{x}_b)^T \Lambda_t (x_b \boxminus \check{x}_b) \\ &+ g_t^T (x_b \boxminus \check{x}_b) + \frac{1}{2}\|f\|^2, \end{aligned} \tag{31}$$

where

$$\Lambda_t = \Lambda_{bb} - \Lambda_{bm}\Lambda_{mm}^{\dagger}\Lambda_{bm}^T \tag{32}$$

$$g_t = g_b - \Lambda_{bm}\Lambda_{mm}^{\dagger} g_m. \tag{33}$$

Equation 32 is the *Schur complement*.

Λ_t can be singular, which is problematic for some optimization methods. One way to address this problem is by computing the eigen-decomposition of Λ_t [8]. By the spectral theorem in linear algebra, Λ_t has an orthonormal eigenbasis. Let r be the rank of Λ_t and the eigen-decomposition of Λ_t be

$$\Lambda_t = UDU^T, \tag{34}$$

Shown in Appendix .1, Equation 31 can be written as the sum of squares:

$$\tilde{c}_{bm}(x_b, \delta_m^*) = \frac{1}{2}\|D^{1/2}U^T(x_b \boxminus \check{x}_b) + D^{-1/2}U^T g_t\|^2. \tag{35}$$

The eigen-decomposition is relatively expensive and iterative. One can instead compute a Cholesky factorization with approximately $n^3/3$ floating point operations $\Lambda_t = P^T LDL^T P$, where

P is a permutation matrix, L is unit lower triangulator, and D is diagonal. Supposing this yields D with positive entries on the diagonal, one may represent the marginalization prior cost with the Cholesky factor $S = P^T L D^{1/2}$ as

$$\tilde{c}_{bm}(x_b, \delta_m^*) = \frac{1}{2}\|S^T(x_b \boxminus \check{x}_b) + S^{-1}g_t\|^2. \tag{36}$$

For ill-conditioned systems, the factorization not exist, and if it does, one can obtain D with nonpositive entries on the diagonal. Depending on the details of the factorization, simply clamping the values of D can correspond to making a large modification to Λ_t. One solution is to add a multiple of identity to Λ_t until the factorization yields $D > 0$ [17]. However, this incurs a large computational expense and effectively injects a spurious measurement.

Various modified Cholesky algorithms are attractive alternatives [18, 19, 20, 21]. These algorithms have approximately the same computational cost as standard Cholesky factorization, are explicitly designed to minimize perturbations of the matrix, and avoid restarting factorization. Our implementation includes various options, which are revisited in section 5.4.

Relating the result to the original objective, one can summarize the steps in eliminating x_m as follows

$$\begin{aligned}\min_{x_m} C(x) =& C_{br}(x_b, x_r) + \min_{\delta_m} c_{bm}(x_b, \delta_m) &(37)\\ \approx& C_{br}(x_b, x_r) + \min_{\delta_m} \tilde{c}_{bm}(x_b, \delta_m) &(38)\\ =& C_{br}(x_b, x_r) + \\ & \frac{1}{2}\|S^T(x_b \boxminus \check{x}_b) + S^{-1}g_t\|^2. &(39)\end{aligned}$$

A geometric view of the cost is shown in Figure 1. The new objective may be re-linearized with respect to x_b using a new linearization point for x_b and x_r, denoted $(\check{x}'_b, \check{x}'_r)$. We can obtain the following linearized objective for subsequent optimization:

$$\begin{aligned}C'(\delta_b, \delta_r) = & C_{br}(\check{x}'_b \boxplus \delta_b, \check{x}_r \boxplus \delta_r) \\ & + \frac{1}{2}\|S^T(((\check{x}'_b \boxplus \delta_b) \boxminus \check{x}_b) + (S^T)^{-1}g_t\|^2. \end{aligned} \tag{40}$$

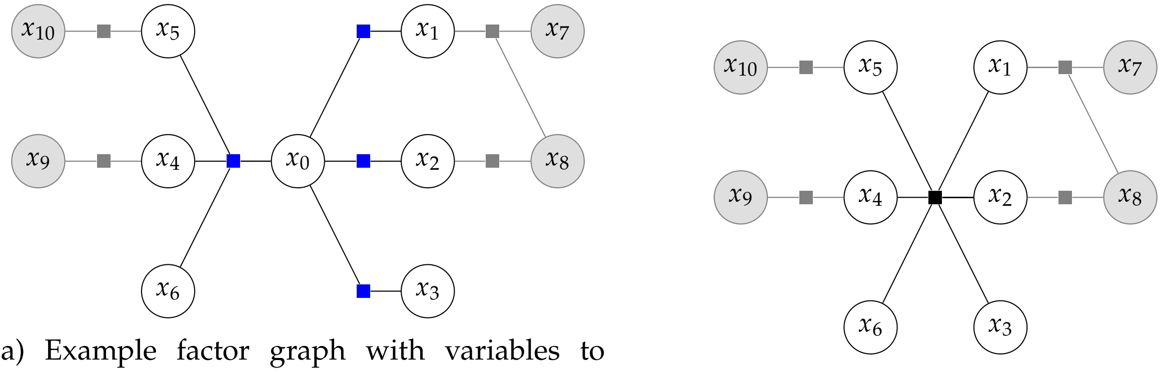
A graphical view of the cost function for the marginalization prior is shown in Figure 2. This requires implementing the Jacobian

$$\frac{\partial}{\partial \delta_b}(\check{x}'_b \boxplus \delta_b) \boxminus \check{x}_b. \tag{41}$$

Fortunately, this Jacobian can be completely encapsulated as an additional manifold operation along with 1, 2, and the $\boxplus$ Jacobian. Correspondingly, our implementation requires that this Jacobian be implemented in the manifold class used for any variable that appears in the Markov blanket. We provide common instances with analytical formulas along with a generic alternative based on automatic differentiation.

3.3 Marginalization via the QR factorization

An algebraically equivalent procedure involves the QR factorization of the Jacobian. We analyze the procedure in the language of $\boxplus$-manifolds and a present an alternative proof using pseudoin-



(a) Example factor graph with variables to marginalize $x_m = (x_0)$, their Markov blanket $x_b = (x_1, x_2, \ldots, x_6)$, and remaining variables $x_r = (x_7, \ldots, x_{10})$. Residual functions are represented as square factor nodes.

(b) Factor graph after eliminating x_0. The marginalized variable and all factors that involve it are replaced with a single factor connected to the Markov blanket.

Figure 1: Elimination of variables is local in the graph, affecting the variables to marginalize and ones related to them by residual functions, their Markov blanket.

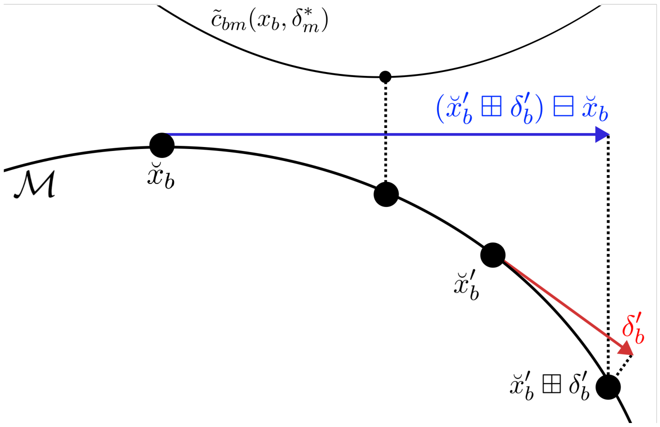

$\tilde{c}_{bm}(x_b, \delta_m^*)$
$(\breve{x}_b' \boxplus \delta_b') \boxminus \breve{x}_b$
$\mathcal{M}$
$\breve{x}_b$
$\breve{x}_b'$
δ_b'
$\breve{x}_b' \boxplus \delta_b'$

Figure 2: The cost for the marginalization prior depends on the linearization point $\breve{x}_b$ at the time of marginalization. Superimposed above, the cost associated with the marginalization prior $\tilde{c}_{bm}$ is a quadratic function of the deviation of the Markov blanket from $\breve{x}_b$ (blue). Subsequent optimizations compute a step in the tangent space δ_b' (red) from future linearization points $\breve{x}_b'$, which must be mapped to the original tangent space (blue).

verse properties. The linearization in Equation 15 can be written compactly as

$$\tilde{c}_{bm}(x_b, \delta_m) = \frac{1}{2} \sum_{i \in \partial x_m} \|f_i(x_b, \breve{x}_m \boxplus \delta_m)\|^2 \tag{42}$$

$$\approx \frac{1}{2} \|f + \begin{bmatrix} J_m & J_b \end{bmatrix} \begin{bmatrix} \delta_m \\ x_b \boxminus \breve{x}_b \end{bmatrix} \|^2. \tag{43}$$

Here, J_b and J_m need not be full-rank. Let r be the rank of J_m, so that $r \leq m$, and let N be the dimension of the residual f. [10] provide a specialized QR factorization

$$\begin{bmatrix} J_m & J_b \end{bmatrix} = QR, \tag{44}$$

$$R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix}, \tag{45}$$

where Q is an orthogonal matrix, R is upper triangular, $R_{11} \in \mathbb{R}^{r \times m}$, $R_{22} \in \mathbb{R}^{N-r \times b}$, and $R_{12} \in \mathbb{R}^{r \times b}$. Define the modified residual

$$f' \triangleq Q^T f = \begin{bmatrix} f_1' \\ f_2' \end{bmatrix}, \tag{46}$$

where $f_1 \in \mathbb{R}^r$ and $f_2 \in \mathbb{R}^{N-r}$.

In Appendix .2, we show that the cost for the marginalization prior with δ_m minimized out is

$$\min_{\delta_m} \tilde{c}_{bm}(x_b, \delta_m) = \frac{1}{2} \|f_2' + R_{22}(x_b \boxminus \breve{x}_b)\|^2. \tag{47}$$

The result in Equation 47 is associated with a QR factorization where $R = R_{22}$ and with the new residual f_2'. Thus, marginalization via QR factorization requires slicing the matrix R. In addition, [10] proves that the cost is the same as in Equation 31.

4 Additional implementation details

4.1 Generic root-shift operation

In the special cases of pose estimation, it can be advantageous to reparameterize variables prior to marginalization so that poses are represented in a relative coordinate frame. Generalizing this so-called "root-shift operation" follows the spirit of this report, so we describe it Appendix .3. We have observed that choices in root-shift procedures are highly detailed and problem-dependent, so we include it only in our implementation but not in the numerical experiments to follow.

4.2 Choice of linearization points

There is a distinction made between "local" and "global" linearization points. The former are minimizers of Equation 9, and the latter are the minimizers of the original problem in Equation 5. In [9], Eckenhoff et al. use local linearization points which yield $g_t = 0$, allowing simplifications of the cost function for the marginalization prior. When the terms involving g_t are included, superior results can be obtained with global linearization points [22]. Both works describe advantages to relative-frame linearization points over global-frame linearization points. An additional consideration is statistical consistency, which can require alternative choices of linearization points for computing "first-estimates Jacobians" [23, 1, 24]. To support these requirements, our implementation can make use of user-provided linearization points. In this report, all results use global linearization points.

4.3 Outliers and Robust Loss Functions

Many relevant applications must cope with outlier measurements (e.g. feature reprojections). This is commonly addressed by augmenting the cost in Equation 5 with robust loss functions. The modified objective can be written as

$$C_{\text{robust}}(x) = \frac{1}{2} \sum_i \rho_i(\|f_i(x_{i_1}, \ldots, x_{i_k})\|^2), \tag{48}$$

where ρ_i is a robust loss function, such as the Huber loss. The quadratic approximation in Equation 23 must be modified to account for the high degree of nonlinearity introduced. Gauss-Newton optimization addresses the same problem in computing quadratic approximations of the objective. A common approach is described in [25]. Following this approach, the residuals and Jacobians are weighted before the Schur complement, resulting in a quadratic objective.

4.4 Intra-clique factors

As discussed in [8], intra-clique factors, which are existing terms involving the Markov blanket variables, may be merged with the marginalization prior. Our implementation offers this option.

4.5 Sparsification

Marginalization results in a denser graph as can be seen in Figure 1. Previous work on sparsification can help address this problem and can be considered complementary to this work [8, 9].

5 Example: GPS/INS with fixed-lag smoothing

To illustrate the convenience of prototyping with our implementation, we demonstrate an implementation in Ceres for an estimator fusing measurements from the global positioning system (GPS) in an inertial navigation system (INS). The problem formulation and implementation is intended to be as simple as possible to demonstrate prototyping usage.

A state $x_t = (R_t, p_t, v_t, b_t^a, b_t^g)$ is defined at every GPS measurement consisting of a world-frame rotation $R_t \in SO(3)$, translation $p_t \in \mathbb{R}^3$, and translational velocity $v_t \in \mathbb{R}^3$; accelerometer bias $b_t^a \in \mathcal{R}^3$, and gyroscope bias $b_t^g \in \mathbb{R}^3$. The world-frame is gravity aligned, and the gravity magnitude is fixed to the ground truth. Our estimator is a fixed-lag smoother with lag parameter L, which provides estimates of a sliding window of recent states $x_{t-L}, x_{t-L+1}, ..., x_t$ conditional on measurements up to time step t.

5.1 Model and Ceres cost functions

Since the choice of cost functions for the problem is not the focus of this work, we provide a short summary and refer to our public implementation. The GPS measurements are modeled as

$$z_t = Hx_t + \epsilon_t, \tag{49}$$

where ϵ_t is a zero-mean additive Gaussian white-noise process with a scalar covariance at each time step. For simplicity, the accelerometer is assumed to be coincident with the GPS receiver at time time step t, both having earth-referenced positions as direct state (position) observations Hx_t.

Inertial measurement unit (IMU) measurements are incorporated in a simple cost function based on simple on-manifold zeroth order accelerometer- and gyro-bias-corrected IMU integration between subsequent states, each state corresponding to one GPS measurement. In addition, a simple cost function to impose slowly-varying biases is included. Both cost functions only depend on subsequent states coinciding with GPS measurements. Under the ideal conditions of noiseless data, perfect states, and the continuous limit, the integration satisfies

$$\begin{align} R_{t+1} &= f_r(x_t) \tag{50} \\ p_{t+1} &= f_p(x_t) \tag{51} \\ v_{t+1} &= f_v(x_t). \tag{52} \end{align}$$

To account for errors in integration, we use a simple cost function

$$\begin{align} &\sum_t \lambda_r \| f_r(x_t) \boxminus R_{t+1} \|_2^2 + \lambda_p \| f_p(x_t) \boxminus p_{t+1} \|_2^2 \notag \\ &\quad + \lambda_v \| f_v(x_t) \boxminus v_{t+1} \|_2^2 \notag \\ &\quad + \lambda_{b^g} \| b^g_{t+1} - b^g_t \|^2 + \lambda_{b^a} \| b^a_{t+1} - b^a_t \|^2 \tag{53} \\ &\quad + \lambda_z \| z_t - Hx_t \|^2, \tag{54} \end{align}$$

where the inverse covariances $\lambda_r, \lambda_p, \lambda_v, \lambda_{b^g}, \lambda_z$ are scalars.

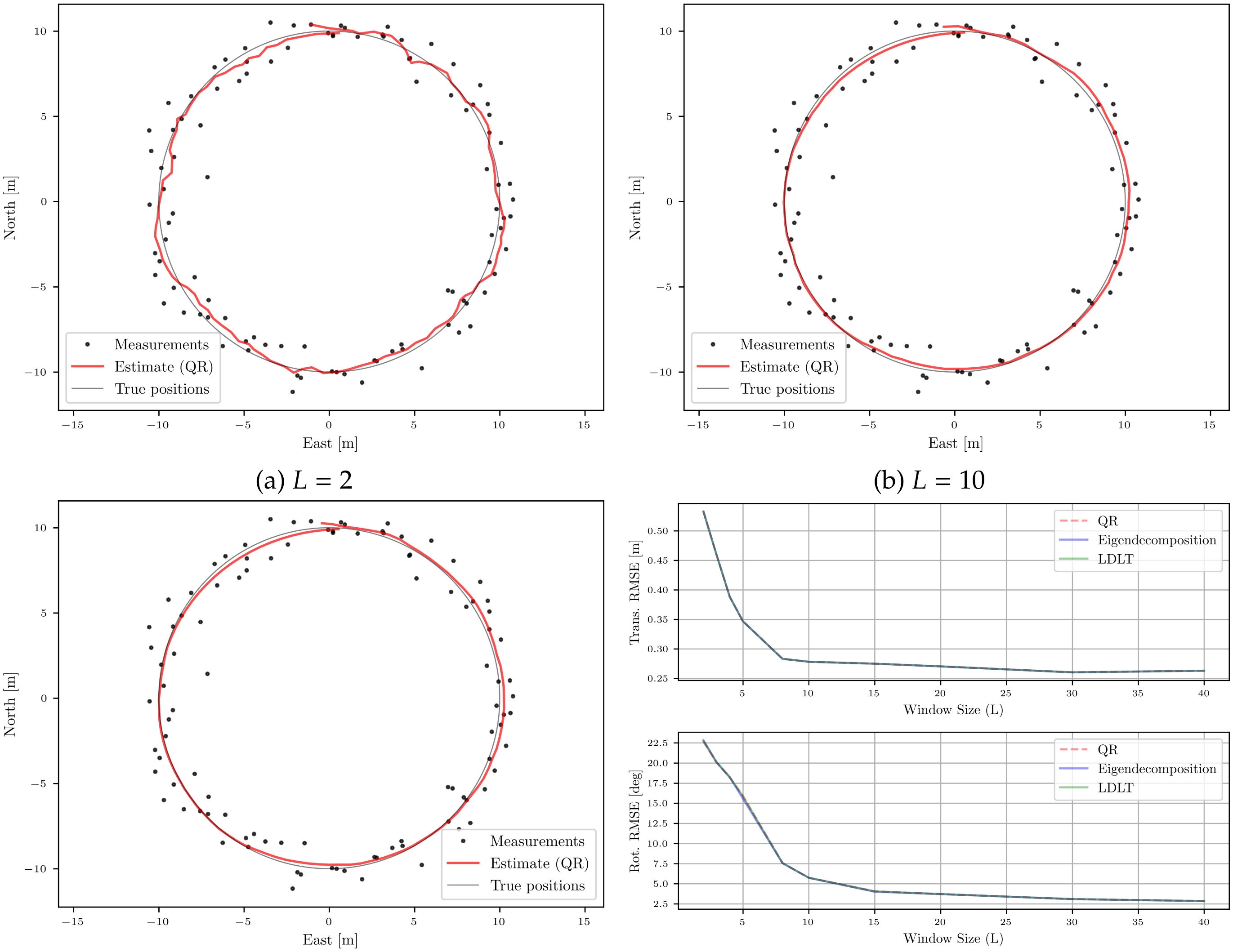
5.2 Main loop in Ceres

To demonstrate a typical workflow with our implementation, we show only the loop over GPS measurements used in the GPS/INS example. Given any problem consisting of variables that implement 41 in addition to the basic operators, only the call to MarginalizeOutVariables is required to marginalize out an arbitrary set of variables. Under the hood, this call performs a local search in the graph around the variables to marginalize out, constructs a small problem involving x_b and x_m, runs any of the previous procedures to compute a new cost function involving x_b, and finally adds a new cost function to the graph.

```
ceres::Problem problem;
vector<Vector> states(gpsMeas.size());
deque<State*> problem_states;
for (size_t t = 0; t<gpsMeas.size(); ++t){
  // Translation state.
  problem.AddParameterBlock(states[t].t,9);
  // Rotation state.
  problem.AddParameterBlock(states[t].R,9);
  problem.SetManifold(states[t].R,
    new MarginalizableSO3Manifold());
  // Accel bias, gyro bias, velocity states.
  problem.AddParameterBlock(states[t].vb,9);
  problem_states.push_back(&states[t]);
  // Marginalization for lag L.
  if (problem_states.size() > L) {
      MarginalizeOutVariables(
        MarginalizationOptions(),
        {problem_states.front()},
        &problem);
      problem_states.pop_front();
    }
    // IMU cost function
    if (t > 0) {
      auto* imu_cost =
          new ImuErrorCostFunction(
            imuParams, imuMeas[t-1],
            gpsMeas[t-1].Time,
            gpsMeas[t].Time);
      problem.AddResidualBlock(
        imu_cost,  /*loss_function=*/nullptr,
        states[t-1].R, states[t-1].t,
        states[t-1].vb,
        states[t].R, states[t].t,
        states[t].vb);
    }
    // GPS cost function
    auto* gps_cost = new GpsCostFunction(
        pos_info, gpsMeas[t].XYZ);
    problem.AddResidualBlock(
        gps_cost,/*loss_function*/nullptr,
        states[t].t);
    // Solve
    SolveOptimizationProblem(&problem);
    // Predict the next state.
    if (t + 1 < gpsMeas.size())
    {
      IntegrateImuMeasurements(
        imuMeas[t], gpsMeas[t].Time,
        states[t], gpsMeas[t+1].Time,
        states[t+1]);
    }
  }
```

5.3 Evaluation with synthetic data

Measurements and ground truth positions were synthesized from a circular trajectory with constant linear velocity and a periodically varying rotation and radius 10 meters. The accelerometer, gyroscope, and GPS receiver were coincident. Synthetic, noisy accelerometer and gyroscope measurements were synthesized at a synchronized 1 kHz rate with additive Gaussian white noise with standard deviation 0.1 m/sec^2 and 0.1 rad/sec along with constant biases of $10^{-3}(1,1,1)$ m/sec^2 and $10^{-2}(1,1,1)$ rad/sec respectively. Synthetic, noisy GPS measurements were generated as Gaussian white noise-corrupted versions of the ground truth position with standard deviation 2



(a) $L = 2$

(b) $L = 10$

(c) $L = 20$

(d) Translation and rotation error v. L.

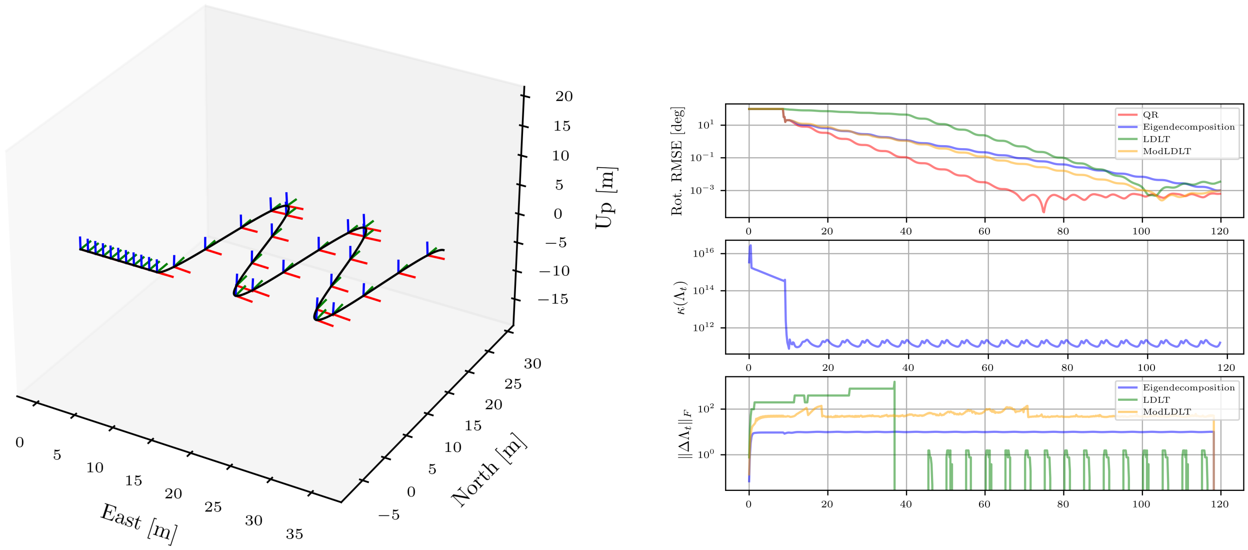
Figure 3: Results for GPS/INS with fixed-lag smoothing are shown with various L. Results with different factorizations had negligible differences.

meters. Cost function parameters were $\lambda_{b^a} = \lambda_{b^g} = 10^5$, $\lambda_v = 10$, $\lambda_r = 10^4$, $\lambda_p = 10^2$, $\lambda_z = 1$. 5 Levenberg–Marquardt iterations were used in Ceres with default parameters.

To demonstrate the ability of our implementation to handle rank-deficient Jacobians in marginalization, no priors were used, including any prior on the initial state. A maximum window size of L was maintained by marginalizing out the oldest state. Propagation from the latest state was used to initialize new ones.

Our public Ceres-based implementation offers two alternative implementation options based on the Schur complement and QR factorization described in Sections 3.2 and 3.3. The Ceres library and our implementation only support double precision. A detailed comparison of the two implementations and their numerical properties is outside the scope and well-described in literature. We compare the results from both here to demonstrate that both handle rank-deficiency and show nearly identical results in a problems that are otherwise sufficiently well-conditioned.

One can investigate tradeoffs in varying L and computing the overall accuracy of all smoothed state estimates x_{t-L} conditional on measurements up to time step t. This accuracy-lag tradeoff is illustrated in Figure 3.



(a) Ground truth trajectory from a synthetic dataset with sinusoidal portion truncated for clarity. The body transitions from constant world-frame translational velocity to periodic rotation and translation as all states become observable.

(b) Naive LDL^T factorization and failure handling shows slower recovery due to handling of failures, while other factorizations show faster recovery. As states become observable, the condition number $\kappa(\Lambda_t)$ drops off. The perturbations to Λ_t reflected in $\|\Lambda_t\|_F$ correspond to the various recovery rates.

5.4 Numerical stability

Section 3.2 described marginalization via the QR factorization and Schur complement. The Schur complement can be implemented with the eigendecomposition, the LDL^T factorization, or a modified LDL^T factorization (permutations omitted for brevity), while the latter two are the least computationally expensive. We illustrate tradeoffs in numerical accuracy and efficiency observed in marginalization with these four approaches in an ill-conditioned GPS/INS problem.

In the problem we construct, the state is the same as the one in Section 5.3. All GPS and IMU data are noiseless synthetic data. GPS and IMU data were sampled at 5Hz and 10kHz. A weak initial prior on the state was included with inverse covariance 10^{-13}. The inverse covariance used for GPS measurements was 10^9. For IMU terms, $\lambda_{b^a} = \lambda_{b^g} = 10^9$ and $\lambda_v = \lambda_r = \lambda_p = 10$. A single Gauss-Newton iteration was used at each time step.

The body initially follows a trajectory with a 100 degree rotation error and constant world-frame translational velocity and zero angular velocity along a line. Rotation is not observable in this period. After 10 seconds, it undergoes sinusoidal translation and rotation as shown in Figure 4a, when all states become observable.

Conventional LDL^T factorization was considered successful if all diagonal entries of D were nonnegative. The implementation used pivoting [26]. Failures were handled by re-attempting factorization of Λ_t after adding a multiple of identity as described in [17]. Our modified Cholesky factorization is from [18], which we refer to as “ModLDLT". For the eigendecomposition, we retain nonzero eigenvalues. For all methods, Λ_m^{-1} was computed with an LDL^T factorization that succeeded in all attempts.

Figure 4b shows that the rotation error over time for the four methods. For all methods, translation error was negligible ($< 10^{-10}$ meters). After 10 seconds, the rotation error decays, and the decay is gradual for LDL^T. Correspondingly, the condition number computed from the Jacobian $\begin{bmatrix} J_b & J_m \end{bmatrix}$ when the eigendecomposition is used drops.

For such an ill-conditioned problem, LDL^T must perturb Λ_t by adding a multiple of identity, effectively injecting a spurious measurement of zero error. As expected, zero-error measurements slow recovery. Any factorization used in the Schur complement make a perturbation of Λ_t, denoted $\Delta\Lambda_t$, and their size as measured by $\|\Delta\Lambda_t\|_F$ shows that LDL^T introduces the largest perturbation. This corroborates the intuition that spurious information can be injected by a common failure handling strategy. The modified Cholesky factorization achieves the lowest overall computational cost and requires smaller perturbations in the worst case.

6 Conclusion

In this report, we presented a generic formulation of marginalization on $\boxplus$-manifolds. We have described a common framework for various algorithmic choices, which were compared in numerical experiments. Translating these ideas to code, we allow users to marginalize out variables with a single line of code in Ceres. Practictioners can implement generic sensor fusion algorithms and experiment with various algorithmic choices in a natural way.

References

[1] Stefan Leutenegger et al. "Keyframe-based visual–inertial odometry using nonlinear optimization". In: *The International Journal of Robotics Research* 34.3 (2015), pp. 314–334.

[2] Jakob Engel, Vladlen Koltun, and Daniel Cremers. "Direct sparse odometry". In: *IEEE transactions on pattern analysis and machine intelligence* 40.3 (2017), pp. 611–625.

[3] Vladyslav Usenko et al. "Visual-inertial mapping with non-linear factor recovery". In: *IEEE Robotics and Automation Letters* 5.2 (2019), pp. 422–429.

[4] Tong Qin, Peiliang Li, and Shaojie Shen. "Vins-mono: A robust and versatile monocular visual-inertial state estimator". In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 1004–1020.

[5] Christoph Hertzberg et al. "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds". In: *Information Fusion* 14.1 (2013), pp. 57–77.

[6] Tom L Koller and Udo Frese. "The Interacting Multiple Model Filter on Boxplus-Manifolds". In: *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE. 2020, pp. 88–93.

[7] Dongjiao He, Wei Xu, and Fu Zhang. "Kalman Filters on Differentiable Manifolds". In: *arXiv preprint arXiv:2102.03804* (2021).

[8] Nicholas Carlevaris-Bianco, Michael Kaess, and Ryan M Eustice. "Generic node removal for factor-graph SLAM". In: *IEEE Transactions on Robotics* 30.6 (2014), pp. 1371–1385.

[9] Kevin Eckenhoff, Liam Paull, and Guoquan Huang. "Decoupled, consistent node removal and edge sparsification for graph-based SLAM". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 3275–3282.

[10] Nikolaus Demmel et al. "Square Root Marginalization for Sliding-Window Bundle Adjustment". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 13260–13268.

[11] Giorgio Grisetti et al. "g2o: A general framework for (hyper) graph optimization". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2011, pp. 9–13.

[12] Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. *Ceres Solver*. Version 2.1. Mar. 2022. URL: https://github.com/ceres-solver/ceres-solver.

[13] Frank Dellaert and Chris Beall. "GTSAM 4.0". In: *URL: https://bitbucket. org/gtborg/gtsam* (2017).

[14] Dehann Fourie et al. "Characterizing marginalization and incremental operations on the Bayes tree". In: *Algorithmic Foundations of Robotics XIV: Proceedings of the Fourteenth Workshop on the Algorithmic Foundations of Robotics 14*. Springer. 2021, pp. 227–242.

[15] Hayk Martiros et al. "SymForce: Symbolic Computation and Code Generation for Robotics". In: *Proceedings of Robotics: Science and Systems*. 2022. DOI: 10.15607/RSS.2022.XVIII.041.

[16] Frank Dellaert, Michael Kaess, et al. "Factor graphs for robot perception". In: *Foundations and Trends® in Robotics* 6.1-2 (2017), pp. 1–139.

[17] Stephen Wright, Jorge Nocedal, et al. "Numerical optimization". In: *Springer Science* 35.67-68 (1999), p. 7.

[18] PE Gill, W Murray, and Margaret Wright. *Practical optimization*. Academic Press, 1981.

[19] Sheung Hun Cheng and Nicholas J Higham. "A modified Cholesky algorithm based on a symmetric indefinite factorization". In: *SIAM Journal on Matrix Analysis and Applications* 19.4 (1998), pp. 1097–1110.

[20] Robert B Schnabel and Elizabeth Eskow. "A revised modified Cholesky factorization algorithm". In: *SIAM Journal on optimization* 9.4 (1999), pp. 1135–1148.

[21] Haw-ren Fang and Dianne P O'leary. "Modified Cholesky algorithms: a catalog with new approaches". In: *Mathematical Programming* 115.2 (2008), pp. 319–349.

[22] Daniel Wilbers, Lars Rumberg, and Cyrill Stachniss. "Approximating marginalization with sparse global priors for sliding window SLAM-graphs". In: *2019 Third IEEE International Conference on Robotic Computing (IRC)*. IEEE. 2019, pp. 25–31.

[23] Guoquan P Huang, Anastasios I Mourikis, and Stergios I Roumeliotis. "A first-estimates Jacobian EKF for improving SLAM consistency". In: *Experimental Robotics*. Springer. 2009, pp. 373–382.

[24] Joel A Hesch et al. "Observability-constrained vision-aided inertial navigation". In: *University of Minnesota, Dept. of Comp. Sci. & Eng., MARS Lab, Tech. Rep* 1 (2012), p. 6.

[25] Bill Triggs et al. "Bundle adjustment—a modern synthesis". In: *International workshop on vision algorithms*. Springer. 1999, pp. 298–372.

[26] Gaël Guennebaud, Benoit Jacob, et al. "Eigen". In: *URl: http://eigen. tuxfamily. org* 3 (2010).

.1 Marginalization prior cost via the Schur complement

Here, we prove equality of the conversion from the cost in Equation 31 and the sum-of-squares cost in Equation 35.

Let $\delta_b = x_b \boxminus \check{x}_b$. Substituting Equations 24 - 28 into Equation 31 yields

$$\begin{aligned}\tilde{c}_{bm}(x_b, \delta_m^*) =& \frac{1}{2}\|f\|^2 + f^T P J_b \delta_b \\ &+ \frac{1}{2}\delta_b^T J_b^T P J_b \delta_b,\end{aligned} \tag{55}$$

where

$$P = I - J_m(J_m^T J_m)^{-1} J_m^T. \tag{56}$$

We can see that $\tilde{c}_{bm}$ has the form of a nonlinear least squares cost function. Specifically, there exists an A with linearly independent columns and b such that

$$\tilde{c}_{bm}(x_b, \delta_m^*) = \frac{1}{2}\delta_b^T A^T A \delta_b + b^T A \delta_b + \frac{1}{2}\|b\|^2. \tag{57}$$

Consider the singular value decomposition

$$A = U_A \Sigma_A V_A^T, \tag{58}$$

where Σ_A contains the nonzero singular values on the diagonal and

$$U_A^T U_A = U_A U_A^T = I \tag{59}$$

$$V_A^T V_A = I \tag{60}$$

With access to $A^T A = V_A \Sigma_A^2 V_A^T$, we can compute Σ_A^2 and V_A. Since $g_t = A^T b$, we have

$$\begin{aligned}\tilde{c}_{bm}(x_b, \delta_m^*) =& \frac{1}{2}\delta_b^T V_A \Sigma U^T U \Sigma V_A^T \delta_b \\ &+ g_t^T V_A \Sigma_A^{-1} \Sigma_A V_A^T \delta_b + \frac{1}{2} g_t^T V_A \Sigma^{-2} V_A^T g_t\end{aligned} \tag{61}$$

The first terms in Equation 31 and Equation 61 can be shown to match.

$$\frac{1}{2}\delta_b^T V_A \Sigma_A U^T U \Sigma_A V_A^T \delta_b = \frac{1}{2}\delta_b^T V_A \Sigma_A^2 V_A^T \delta_b \tag{62}$$

$$= \frac{1}{2}\delta_b^T A^T A \delta_b \tag{63}$$

$$= \frac{1}{2}\delta_b^T \Lambda_t \delta_b \tag{64}$$

To show that the second term matches, we substitute $g_t = A^T b$, the SVD for A, and Equation 60

$$g_t^T V_A \Sigma_A^{-1} \Sigma_A V_A^T \delta_b = b^T A V_A \Sigma_A^{-1} \Sigma_A V_A^T \delta_b \tag{65}$$

$$= b^T U_A \Sigma_A V_A^T V_A \Sigma_A^{-1} \Sigma_A V_A^T \delta_b \tag{66}$$

$$= b^T U_A \Sigma_A V_A^T \delta_b \tag{67}$$

$$= b^T A \delta_b \tag{68}$$

$$= g_t^T \delta_b, \tag{69}$$

To show that the third term matches, the same substitutions can be used in addition to Equation 59.

$$\begin{aligned}
&\frac{1}{2} g_t^T V_A \Sigma_A^{-2} V_A^T g_t \\
&= \frac{1}{2} b^T A V_A \Sigma_A^{-2} V_A^T A^T b && (70)\\
&= \frac{1}{2} b^T U_A \Sigma_A V_A^T V_A \Sigma_A^{-2} V_A^T V_A \Sigma_A U_A^T b && (71)\\
&= \frac{1}{2} b^T U_A \Sigma_A \Sigma_A^{-2} \Sigma_A U_A^T b && (72)\\
&= \frac{1}{2} b^T U_A U_A^T b && (73)\\
&= \frac{1}{2} \|b\|^2 && (74)
\end{aligned}$$

.2 Marginalization prior cost via QR factorization

Substituting Equations 44 and 46 into Equation 43, we obtain

$$\begin{aligned}
\tilde{c}_{bm}(x_b, \delta_m) &= \frac{1}{2} \| f + QR \begin{bmatrix} \delta_m \\ (x_b \boxminus \breve{x}_b) \end{bmatrix} \|^2 && (75)\\
&= \frac{1}{2} \| Q^T f + \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix} \begin{bmatrix} \delta_m \\ (x_b \boxminus \breve{x}_b) \end{bmatrix} \|^2 && (76)\\
&= \frac{1}{2} \| Q^T f + \begin{bmatrix} R_{11} \\ 0 \end{bmatrix} \delta_m + \begin{bmatrix} R_{12} \\ R_{22} \end{bmatrix} (x_b \boxminus \breve{x}_b) \|^2 && (77)\\
&= \frac{1}{2} \| Q^T f + \begin{bmatrix} R_{11} \\ 0 \end{bmatrix} \delta_m + \begin{bmatrix} R_{12} \\ R_{22} \end{bmatrix} (x_b \boxminus \breve{x}_b) \|^2 && (78)
\end{aligned}$$

The pseudoinverse provides a solution for δ_m that minimizes $\tilde{c}_{bm}$.

$$\begin{aligned}
\delta_m^* &= \arg\min_{\delta_m} \tilde{c}_{bm}(x_b, \delta_m) && (79)\\
&= -R_{11}^{\dagger}(f_1' + R_{12}(x_b \boxminus \breve{x}_b)). && (80)
\end{aligned}$$

Substituting this into Equation 78 and some rearrangement yields

$$\begin{aligned}
\tilde{c}_{bm}(x_b, \delta_m^*) &= \frac{1}{2} \| \begin{bmatrix} f_1' \\ f_2' \end{bmatrix} - \begin{bmatrix} R_{11} R_{11}^{\dagger}(f_1' + R_{12}(x_b \boxplus \breve{x}_b)) \\ 0 \end{bmatrix} \\
&+ \begin{bmatrix} R_{12} \\ R_{22} \end{bmatrix} (x_b \boxminus \breve{x}_b) \|^2 && (81)\\
&= \frac{1}{2} \| \begin{bmatrix} f_1' \\ f_2' \end{bmatrix} - \begin{bmatrix} f_1' + R_{12}(x_b \boxplus \breve{x}_b) \\ 0 \end{bmatrix} \\
&+ \begin{bmatrix} R_{12} \\ R_{22} \end{bmatrix} (x_b \boxminus \breve{x}_b) \|^2 && (82)\\
&= \frac{1}{2} \| \begin{bmatrix} 0 \\ f_2' \end{bmatrix} + \begin{bmatrix} 0 \\ R_{22} \end{bmatrix} (x_b \boxminus \breve{x}_b) \|^2 && (83)\\
&= \frac{1}{2} \| f_2' + R_{22}(x_b \boxminus \breve{x}_b) \|^2. && (84)
\end{aligned}$$

In Equation 82, we used the fact that R_{11} is has linearly independent rows, so $R_{11}^{\dagger}$ satisfies $R_{11}R_{11}^{\dagger} = I$.

.3 Generic root-shift operation

Consider the case where all variables are poses belonging to a group G. Let $x_{wk} \in G$ be the original global-frame representation for node k in the elimination clique. There is freedom in choosing the coordinate system on which they are based. As discussed in related work, the global coordinate frame can be a poor choice because a stable global-frame linearization point may not exist [8, 9]. Thus, prior to marginalization, it can be advantageous to reparameterize any poses expressed in the global frame in a local and stable coordinate frame. We provide an alternative generic description of the procedure described in previous work.

Choosing node 1 as the node defining the new local coordinate system, the reparameterized variables are defined as

$$\begin{bmatrix} x_{w1} \\ x_{12} \\ x_{13} \\ \vdots \end{bmatrix} \triangleq r(\begin{bmatrix} x_{w1} \\ x_{w2} \\ x_{w3} \\ \vdots \end{bmatrix}) = \begin{bmatrix} x_{w1} \\ x_{w1}^{-1} \circ x_{w2} \\ x_{w1}^{-1} \circ x_{w3} \\ \vdots \end{bmatrix} \tag{85}$$

Consider a residual term in the elimination clique f, which may already be a function of x_{wk} for all $k \geq 1$. As part of this operation, f is replaced with the residual

$$f'(x_{w1}, x_{12}, x_{13}, \dots) \triangleq f(x_{w1}, x_{w1} \circ x_{12}, x_{w1} \circ x_{13}, \dots) \tag{86}$$

Applying the chain rule, the derivatives of f' with respect to the reparameterized variables are

$$\frac{\partial f'}{\partial x_{w1}} = \frac{\partial f}{\partial x_{w1}} + \sum_{k \geq 2} \frac{\partial f}{\partial x_{wk}} \frac{\partial}{\partial x_{w1}}(x_{w1} \circ x_{1k}) \tag{87}$$

$$\frac{\partial f'}{\partial x_{1k}} = \frac{\partial f}{\partial x_{wk}} \frac{\partial}{\partial x_{1k}}(x_{w1} \circ x_{1k}) \tag{88}$$

The result expression shows that the derivatives of the new residual are functions of the derivatives of the original residual and the derivatives of the composition operator for the group G. Thus, one can implement the modified residual without knowing any additional information about the original residual. The only group properties that must be implemented to perform this step are only the composition operators, their derivatives, and composition with the inverse of x_{w1}.