A Factorization Approach for Enabling Structure-from-Motion/SLAM Using Integer Arithmetic

Nilesh A. Ahuja
Intel Labs

Mahesh Subedar
Intel Labs

Yeongseon Lee
Intel

Omesh Tickoo
Intel Labs

Abstract

SLAM and SfM algorithms typically involve minimization of a cost-function by non-linear least-squares methods. The matrices involved are typically very poorly conditioned, making the procedure sensitive to numerical precision effects. Ensuring accuracy therefore entails the use of high-precision floating-point data-types for representation and compute. In this paper, a square-root filtering approach to EKF-based SfM is presented and is shown to be capable of operating with lower-precision arithmetic than the EKF, while sacrificing only a little in accuracy. Specifically, we demonstrate a prototype that is capable of operating with integer arithmetic rather than floating-point - the first such implementation to the best of our knowledge. This is important given the increasing need to implement advanced vision-based capabilities on low-power embedded and mobile processors, some of which might not even support floating-point arithmetic for reasons of cost and power. Furthermore, an evaluation of the computational complexity shows that the proposed approach typically requires fewer computations than the EKF in practice, resulting in an algorithm that is both numerically more robust and computationally less intensive.

1. Introduction

Recent years have witnessed a rapid proliferation of power-constrained smart devices such as smartphones, and autonomous drones and robots. Increasingly, such devices are being equipped with advanced computer vision capabilities for scene understanding. A key task is recreating the 3D space around the device, while simultaneously localizing the device in real-time within it, thus enabling it to navigate autonomously. Depending on the specific usage, it might be required to perform either visual-odometry, or structure-from-motion (SfM), or full SLAM. Typically, the platform of implementation of such algorithms is a powerful multi-core CPU capable of supporting high-performance computations with double-precision floating-point arithmetic. Recent works such as PTAM [17, 18] and others [26, 21] attempt to attain real-time operation of localization or SLAM algorithms on smartphone-like devices. They do so by partitioning the problem into a fast problem of tracking that is implemented on a mobile frontend device, and a slow problem of mapping that is implemented on a server backend. More recent work focuses on real-time implementations of visual odometry [11]

However, in all the research mentioned, little attention is paid to the numerical type used for computations. Modern mobile and embedded processors may not always support high-precision arithmetic, or if they do, the performance is severely limited. Therefore, in order to achieve acceptable performance on such processors, algorithms typically have to be implemented using fixed-point (integer) arithmetic. This poses challenges for SLAM algorithms and their variants as these are known to be sensitive to numerical precision errors. In this work, therefore, we demonstrate the benefit of using alternate 'square-root' approaches that are known to be numerically more robust, and therefore allow the use of lower-precision arithmetic. In particular, we apply these principles to Extended Kalman Filter (EKF)-based SfM and use a variant of the Kalman filter, known as the square-root information filter (SRIF)[4]. Unlike the traditional Kalman Filter, which involves computations with a poorly conditioned covariance matrix, square-root filters instead operate on the Cholesky factor of either the covariance matrix or the information matrix, which is the inverse of the covariance matrix. Specifically, it is shown that SRIF enables the implementation of SfM in 32-bit integer (fixed-point) arithmetic - the first such implementation reported to the best of our knowledge. The accuracy of the results using integer arithmetic, though slightly lower than that using floating-point arithmetic, should be tolerable for most applications. Furthermore, the computational cost of the proposed algorithm is less than that of the EKF in practice. This results in significant (typically 2x) performance improvements in real-world implementation on various processors (CPUs, DSPs, or GPUs), owing to the use of lower-precision arithmetic and the reduced op-count.

Although, EKF has been used extensively in SLAM [1, 10, 6, 8, 7, 28], the square-root formulation of the

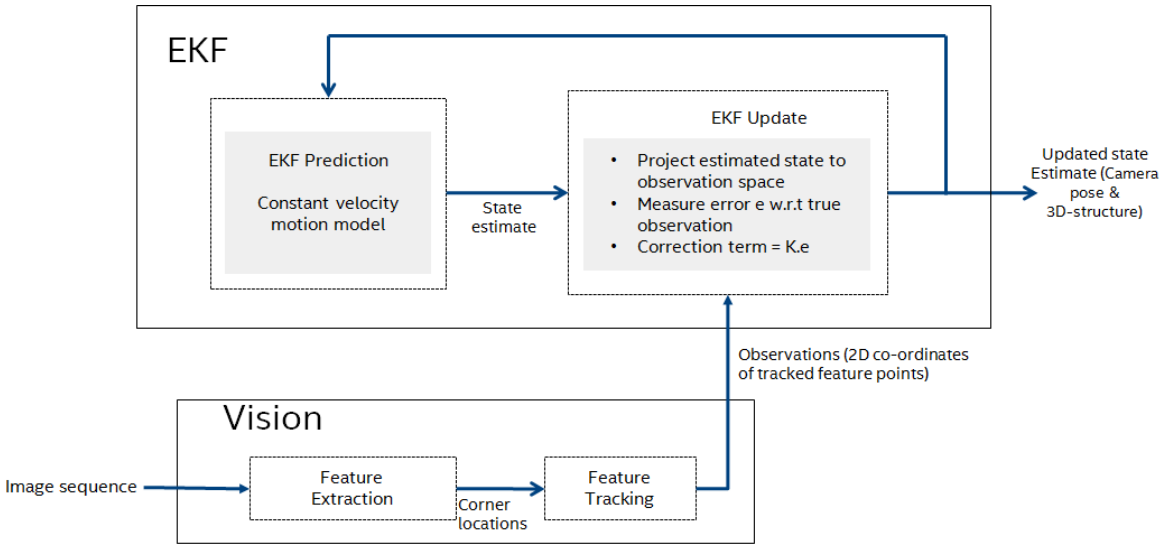

Figure 1. EKF-based Structure-from-Motion.

Kalman filter has found surprisingly little adoption in the vision community, despite its obvious numerical advantages. One of the foremost set of works exploring this for SLAM has been by Dellaert *et al*. in several publications [9, 15, 14]. In these works, square-root filtering is applied to the 'offline SLAM' problem, i.e. the problem of optimally estimating the full set of poses along with all features in the environment from the entire set of observations. The square-root matrix itself is obtained by either the Cholesky factorization of the information matrix $\mathbf{I}(\mathbf{I} = \mathbf{R}^T\mathbf{R})$, or the QR-factorization of the measurement Jacobian $\mathbf{A}(\mathbf{A} = \mathbf{Q}\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix})$. The use of the Cholesky method is recommended in owing to its lower computational cost as well as the ability to leverage sparsity of the information matrix. However, the ability to run the algorithm with lower precision numeric types is not explored; in fact, explicitly forming the information matrix precludes such a possibility because the condition number $\kappa(\mathbf{I})$ of $\mathbf{I}$, is the square of the condition number $\kappa(\mathbf{R})$ of $\mathbf{R}$. By contrast, neither the information matrix nor the covariance matrix is explicitly formed in our work. Another point of contrast is that this work focuses on the real-time SfM problem in which the pose- and feature-estimates are updated every time a new frame is received. This implies that the system-state changes at every frame since some features previously observed would no longer be visible while some new features would have been added. An efficient way to deal with this varying state-size is presented.

An outline of the paper is as follows: an overview of a square-root approach to SLAM is presented in Section 2. These principles are then applied to an EKF-based formulation of SfM (Section 3) and the resulting square-root filtering based approach is presented in Section 4. Functional results are presented in Section 5, in which the error caused by use of lower-precision data-types is measured on a test-set. The computational cost of the square-root method, as well as the effective performance attainable in real-world implementations, is explored in Section 6. The trade-off between error and performance is discussed here. Finally, conclusions are presented in Section 7.

2. A Square-Root Approach to SLAM

Solving the SLAM problem consists of estimating the trajectory of an autonomous system (drone, robot, or vehicle) and a map of its environment as it moves around in it. The trajectory is specified by a sequence of poses, $\mathbf{x} = \{\mathbf{x}_i^D\}$, and the map is specified by the set of coordinates, $\mathbf{m} = \{\mathbf{P}_j^W\}$, of the landmarks in the scene. The system captures measurements $\mathbf{z}_k$ of the landmarks. The measurements are related to the map via a measurement function $\mathbf{z}_k = h_k(\mathbf{x}, \mathbf{m})$. The SLAM problem can be formulated as the following optimization problem

$$\mathbf{x}^*, \mathbf{m}^* = \underset{\mathbf{x},\mathbf{m}}{\arg\min}\ C \tag{1}$$

where,

$$\begin{aligned} C &= \sum_k \|\mathbf{z}_k - h_k(\mathbf{x}, \mathbf{m})\|_W^2 \\ &= \sum_k (\mathbf{z}_k - h_k(\mathbf{x}, \mathbf{m}))^T \mathbf{W} (\mathbf{z}_k - h_k(\mathbf{x}, \mathbf{m})), \end{aligned} \tag{2}$$

and $\mathbf{W}$ is appropriately chosen weight matrix (typically the inverse of the covariance matrix). The optimization is typically solved by Gauss-Newton or Levenberg-Marquadt. Assume that an initial estimate $\left(\mathbf{x}^{(0)}, \mathbf{m}^{(0)}\right)$ is known. Then,

a refined estimate is calculated by linearizing the above cost function around this estimate,

$$\begin{aligned} C &= \sum_k \left\| \mathbf{z}_k - h_k(\mathbf{x}^{(0)}, \mathbf{m}^{(0)}) - \mathbf{H}_k \Delta\mathbf{u} \right\|_W^2 \\ &= \sum_k \left\| \mathbf{e}_k - \mathbf{H}_k \Delta\mathbf{u} \right\|_W^2 \end{aligned} \tag{3}$$

where, $\mathbf{e}_k \triangleq \mathbf{z}_k - h_k(\mathbf{x}^{(0)}, \mathbf{m}^{(0)})$, and,

$$\mathbf{H}_k = \begin{bmatrix} \partial h_k / \partial \mathbf{x} \\ \partial h_k / \partial \mathbf{m} \end{bmatrix}, \text{ and }, \Delta\mathbf{u} = \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{m} \end{bmatrix} \tag{4}$$

The minimization of eq. (3), is traditionally attained by solving

$$\left(\sum_k \mathbf{H}_k^T \mathbf{W} \mathbf{H}_k \right) \Delta\mathbf{u} = \sum_k \mathbf{H}_k \mathbf{W} \mathbf{e}_k \tag{5}$$

Direct inversion of the left-hand side is never carried out; instead the system is solved by Cholesky factorization. An alternate way of solving eq. (3) is by QR-factorization as follows:

$$\begin{aligned} C &= \sum_k \left\| \mathbf{e}_k - \mathbf{H}_k \Delta\mathbf{u} \right\|_W^2 \\ &= \sum_k \left\| \mathbf{R}_w \left(\mathbf{e}_k - \mathbf{H}_k \Delta\mathbf{u} \right) \right\|^2 \\ &= \sum_k \left\| \tilde{\mathbf{e}}_k - \tilde{\mathbf{H}}_k \Delta\mathbf{u} \right\|^2 \end{aligned} \tag{6}$$

where, $\mathbf{R}_w^T \mathbf{R}_w = \mathbf{W}, \tilde{\mathbf{e}}_k = \mathbf{R}_w \mathbf{e}_k$, and $\tilde{\mathbf{H}}_k = \mathbf{R}_w \mathbf{H}_k$. Eq. (6) can be rewritten by stacking the various error and Jacobian matrices as,

$$C = \left\| \tilde{\mathbf{e}} - \tilde{\mathbf{H}} \Delta\mathbf{u} \right\|^2, \tag{7}$$

where

$$\begin{aligned} \tilde{\mathbf{e}} &= \left[\tilde{\mathbf{e}}_1^T | \tilde{\mathbf{e}}_2^T | \ldots | \tilde{\mathbf{e}}_n^T \right], \text{ and} \\ \tilde{\mathbf{H}} &= \left[\tilde{\mathbf{H}}_1^T | \tilde{\mathbf{H}}_2^T | \ldots | \tilde{\mathbf{H}}_n^T \right]. \end{aligned} \tag{8}$$

$\tilde{\mathbf{H}}$ can be factorized by QR-factorization as $\tilde{\mathbf{H}} = \mathbf{Q} \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix}$, where $\mathbf{Q}$ is orthonormal, and $\mathbf{R}$ is upper triangular. Note that this $\mathbf{R}$ matrix is also the Cholesky factor of

$$\tilde{\mathbf{H}}^T \tilde{\mathbf{H}} = \sum_k \mathbf{H}_k^T \mathbf{W} \mathbf{H}_k. \tag{9}$$

Since $\mathbf{Q}^T \mathbf{Q} = \mathbf{I}$, multiplying the inner term in eq. (7) by $\mathbf{Q}^T$ yields

$$C = \left\| \mathbf{Q}^T \tilde{\mathbf{e}} - \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix} \Delta\mathbf{u} \right\|^2. \tag{10}$$

The above equation is easily solved since $\mathbf{R}$ is upper-triangular. The benefit of using this formulation is that the product $\tilde{\mathbf{H}}^T \tilde{\mathbf{H}}$ is never explicitly; rather, all the computations involve only its Cholesky factor which significantly increases the numerical stability of the procedure.

3. EKF-based SfM

We now apply the principles described in the previous section to EKF-based structure-from-motion (SfM). The difference from the previous section is that the map and pose estimation is performed at every input frame rather than waiting for all frames and processing in a batch. The state-vector, therefore contains only the current 6-DOF pose of the device (as opposed to the entire trajectory) and the locations of landmarks currently visible (as opposed to the entire map) and updates these values iteratively at every frame. Specifically, the system state, $\mathbf{x}$, is defined thus:

$$\begin{aligned} \mathbf{x}^D &= \left[\mathbf{q}^D \omega^D \mathbf{v}^D \mathbf{p}^D \right] \\ \mathbf{x}^S &= \left[\mathbf{P}_1^W \mathbf{P}_2^W \cdots \mathbf{P}_N^W \right] \\ \mathbf{x} &= \begin{bmatrix} \mathbf{x}^D \\ \mathbf{x}^S \end{bmatrix} \end{aligned} \tag{11}$$

where, $\mathbf{q}^D$ is the orientation of the device w.r.t the world frame represented by a quaternion, ω^D and $\mathbf{v}^D$ are, respectively, the angular and linear velocities of the device, and $\mathbf{p}^D$ is the position of the device. $\mathbf{P}_k^W$ are the coordinates of the k^{th} feature point, represented using a 6-parameter inverse-depth parametrization [6]. The dimension of $\mathbf{x}$ is therefore $13+6N$, where N is the number of feature points. The error covariance matrix is defined as

$$\mathbf{P} = E\left\{ (\mathbf{x} - \hat{\mathbf{x}}) (\mathbf{x} - \hat{\mathbf{x}})^T \right\} \triangleq \begin{bmatrix} \mathbf{P}^{DD} & \mathbf{P}^{DS} \\ \left(\mathbf{P}^{DS}\right)^T & \mathbf{P}^{SS} \end{bmatrix} \tag{12}$$

and the observation vector is defined as

$$\mathbf{z} = \left[\mathbf{p}_1 \mathbf{p}_2 \cdots \mathbf{p}_N \right]^T \tag{13}$$

and has dimension 2N. Here, $\mathbf{p}_k$ is the 2D image coordinate of the k^{th} feature point $\mathbf{P}_k^W$. The EKF estimates the state variables above from a sequence of observations obtained at discrete time instants. The evolution of the state between two successive observations is described by the propagation model. The EKF utilizes the propagation model to predict the system state at the next time instant. The mapping from the state to the observations is described by the measurement model. The measurements are used to update the value of the system state obtained by prediction. The overall flow of the algorithm is shown in Figure 1.

3.1. Vision pipe

The vision pipe takes as input a sequence of images and outputs the measurements or observations needed for

the EKF update step. The observations are the pixel coordinates of the tracked feature points from one frame to the next frame. The first step in this processing chain is to extract robust features points which are repeatable and provide good localization across multiple frames. The extracted features are then represented by descriptors which can uniquely capture the properties of the feature, such as orientation and edge information. In this work, Harris corners [13] were used as feature points and the Oriented FAST and Rotated BRIEF (ORB) [24] feature descriptor is used which is a binary descriptor. The feature tracking step matches features between successive frames by comparing their locations and their descriptors. The distance between binary descriptors is measured using hamming distance. In order to keep one-to-one correspondence between matched feature points, the matches with minimum distance measure are considered.

3.2. Propagation Model and EKF Prediction

The evolution of the system-state is described by the following non-linear continuous-time system model:

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{\mathbf{x}}^D(t) \\ \dot{\mathbf{x}}^S(t) \end{bmatrix} = \begin{bmatrix} f\left(\mathbf{x}^D(t), \mathbf{w}(t)\right) \\ \mathbf{0} \end{bmatrix} \tag{14}$$

where, f is a constant-velocity motion-model as described in [22], $\mathbf{w}(t)$ is the process noise with covariance $\mathbf{Q}(t)$, and $\dot{\mathbf{x}}^S(t) = \mathbf{0}$ simply implies that the coordinates of the feature points in the world frame are fixed. The predicted value of the state at time t_{k+1} is obtained by:

$$\mathbf{x}^D_{k+1|k} \triangleq \hat{\mathbf{x}}^D\left(t_{k+1}\right) = \int_{t_k}^{t_{k+1}} f\left(\mathbf{x}^D(\tau), \mathbf{w}(t)\right) d\tau \tag{15}$$

To compute the covariance matrix at t_{k+1}, the continuous-time model above is linearized and discretized. The linearized continuous-time model for the device state is

$$\delta\dot{\mathbf{x}}^D(t) = \mathbf{F}\delta\mathbf{x}^D(t) + \mathbf{G}\mathbf{w}(t), \tag{16}$$

where the Jacobians $\mathbf{F}$ and $\mathbf{G}$ are defined as

$$\mathbf{F} = \frac{\partial f}{\partial \mathbf{x}}, \text{ and } \mathbf{G} = \frac{\partial f}{\partial \mathbf{w}}, \tag{17}$$

and $\delta\mathbf{x}^D(t) = \left[\left(\delta\theta^D\right)^T \left(\delta\omega^D\right)^T \left(\delta\mathbf{v}^D\right)^T \left(\delta\mathbf{p}^D\right)^T\right]^T$ is the EKF error-state vector. From standard control-theory, the discrete-time covariance propagation equation is given by:

$$\begin{aligned} \mathbf{P}^{DD}_{k+1|k} &= \Phi^D_k \mathbf{P}^{DD}_{k|k} \left(\Phi^D_k\right)^T + \mathbf{Q}_d \\ \mathbf{P}^{DS}_{k+1|k} &= \Phi^D_k \mathbf{P}^{DS}_{k|k} \\ \mathbf{P}^{SS}_{k+1|k} &= \mathbf{P}^{SS}_{k|k} \text{ since } \dot{\mathbf{x}}^S(t) = \mathbf{0} \end{aligned} \tag{18}$$

where, $\Phi^D_k \triangleq \exp\left(\int_{t_k}^{t_{k+1}} F d\tau\right)$ is the state transition matrix, and $\mathbf{Q}_d$ is the discrete-time system covariance noise. $\mathbf{Q}_d$ can be computed from $\mathbf{Q}, \mathbf{F}$, and $\mathbf{G}$ in a simple fashion using the algorithm from Van Loan [20].

3.3. EKF Update

The measurement model gives the mapping between the system state and the observations:

$$\mathbf{z}_{k+1} = h\left(\mathbf{x}_{k+1}\right) + \mathbf{v}_{k+1} \tag{19}$$

Here, $\mathbf{x}_{k+1} \triangleq \mathbf{x}\left(t_{k+1}\right)$, and $\mathbf{v}_{k+1}$ is the measurement noise with covariance $\mathbf{Q}_v$. The H-matrix is defined as

$$\mathbf{H}_k = \frac{\partial \mathbf{z}_k}{\partial \mathbf{x}} \tag{20}$$

Since we use the inverse-depth measurement model as described in [6], the expression for $\mathbf{H}$ is obtained in a similar fashion. The details of the derivation are skipped here for brevity. The steps involved in EKF update are:

$$\begin{aligned} \hat{\mathbf{e}}_{k+1} &= \mathbf{z}_{k+1} - h\left(\hat{\mathbf{x}}_{k+1|k}\right) \\ \mathbf{S}_{k+1} &= \mathbf{H}_{k+1}\mathbf{P}_{k+1|k}\mathbf{H}^T_{k+1} + \mathbf{Q}_v \\ \mathbf{K}_{k+1} &= \mathbf{P}_{k+1|k}\mathbf{H}^T_{k+1}\mathbf{S}^{-1}_{k+1} \\ \hat{\mathbf{x}}_{k+1|k+1} &= \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_{k+1}\hat{\mathbf{e}}_{k+1} \\ \mathbf{P}_{k+1|k+1} &= \mathbf{P}_{k+1|k} - \mathbf{K}_{k+1}\mathbf{H}_{k+1}\mathbf{P}_{k+1|k} \end{aligned} \tag{21}$$

In practice, the covariance matrix $\mathbf{P}$ is often highly ill-conditioned, which results in the solution being highly sensitive to even small numerical errors. More seriously, such small numerical errors can also cause $\mathbf{P}$ to lose its positive definiteness which results in meaningless outputs from the update step.

3.4. Feature update and removal

Each time a new image is received, some new features would have been observed and some features that were being tracked would no longer be present. For the features that are no longer visible, the corresponding entries from the system state are removed. The covariance matrix is also updated by removing the rows and columns corresponding to the eliminated feature. For newly observed features, both the state-vector and the covariance-matrix are augmented with the entries of the newly initialized values, i.e.

$$\mathbf{x}_{k^+} = \begin{bmatrix} \mathbf{x}_{k^-} \\ \mathbf{x}_{new} \end{bmatrix}, \mathbf{P}_{k^+} = \begin{bmatrix} \mathbf{P}_{k^-} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_{new} \end{bmatrix} \tag{22}$$

For notational convenience, the subscript k^- is used to denote the values after removing features that were visible at time $k-1$ but not at time k, and the the subscript k^+ is used to denote the value after augmenting with the new features at time k.

4. SRIF-based SfM

Square-root information filter (SRIF) is an alternate formulation of the Kalman Filter that operates on the Cholesky

factor of the information matrix: $\mathbf{P}^{-1} = \mathbf{R}^T\mathbf{R}$. The condition number of $\mathbf{R}$ is the square-root of that of $\mathbf{P}^{-1}$ (or, equivalently, $\mathbf{P}$). One of the earliest formulations of square-root filtering can be found in [23]. In this work, the conventional Kalman Filter algorithm was reformulated in terms of the square-root of the covariance matrix to improve numeric accuracy and stability; however, this formulation required more computations than the conventional Kalman Filter [16]. An alternate square-root formulation that operates on the square-root of the information matrix was presented in [4], and with greater exposition and detail in [5]. It was shown that this formulation achieved far superior numerical accuracy compared to the conventional Kalman Filter at a modest increase in computational cost. A survey of the numeric aspects of the various square-root formulations has been presented in [27]. Our work largely follows the presentation of [5] such that the predict and update steps of the EKF operate directly on $\mathbf{R}$ without ever having to compute either $\mathbf{P}$ or $\mathbf{P}^{-1}$. The resulting algorithm is, therefore, less sensitive to numerical errors and can consequently operate with lower-precision numerical types. Details are provided in the following subsections.

4.1. SRIF Update

The following treatment is motivated from the material in [5, Chapters 2 and 5]. It can be shown that the EKF update step is equivalent to performing one iteration of Gauss-Newton algorithm on the following cost functional [2]:

$$C = \left(\mathbf{x} - \hat{\mathbf{x}}_{k+1|k}\right)^T \mathbf{P}^{-1}_{k+1|k} \left(\mathbf{x} - \hat{\mathbf{x}}_{k+1|k}\right) + \left(\mathbf{z}_{k+1} - h(\mathbf{x})\right)^T \mathbf{Q}_v^{-1} \left(\mathbf{z}_{k+1} - h(\mathbf{x})\right) \quad (23)$$

Linearizing about $\hat{\mathbf{x}}_{k+1|k}$, and rearranging terms yields

$$C = \left\| \begin{bmatrix} \mathbf{0} \\ \tilde{\mathbf{e}}_{k+1} \end{bmatrix} + \mathbf{A}\Delta\mathbf{x} \right\|^2, \mathbf{A} = \begin{bmatrix} \mathbf{R}_{k+1|k} \\ \mathbf{R}_v\mathbf{H}_{k+1} \end{bmatrix} \quad (24)$$

where, $\mathbf{Q}_v^{-1} = \mathbf{R}_v^T\mathbf{R}_v$, $\tilde{\mathbf{e}}_{k+1} = \mathbf{R}_v\hat{\mathbf{e}}_{k+1}$, and $\Delta\mathbf{x} = \mathbf{x} - \hat{\mathbf{x}}_{k+1|k}$. $\hat{\mathbf{e}}_{k+1}$ and $\mathbf{Q}_v$ are as defined in subsection (3.3). Performing the QR-factorization of $\mathbf{A}$ and using the fact that $\left\|\mathbf{Q}^T\mathbf{y}\right\| = \|\mathbf{y}\|$ since $\mathbf{Q}$ is orthogonal, yields

$$\begin{aligned} C &= \left\| \begin{bmatrix} \mathbf{0} \\ \tilde{\mathbf{e}}_{k+1} \end{bmatrix} + \mathbf{Q} \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix} \Delta\mathbf{x} \right\|^2 \\ &= \left\| \mathbf{Q}^T \begin{bmatrix} \mathbf{0} \\ \tilde{\mathbf{e}}_{k+1} \end{bmatrix} + \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix} \Delta\mathbf{x} \right\|^2 \end{aligned} \quad (25)$$

The above system can easily be solved by back-substitution since $\mathbf{R}$ is an upper triangular matrix. $\Delta\mathbf{x}$ is added to $\hat{\mathbf{x}}_{k+1|k}$ to obtain $\hat{\mathbf{x}}_{k+1|k+1}$. Furthermore, it can be shown that $\mathbf{P}^{-1}_{k+1|k+1} = \mathbf{R}^T\mathbf{R}$, and hence $\mathbf{R}_{k+1|k+1} = \mathbf{R}$. Thus, at no point is it required to explicitly compute the covariance matrix and it is sufficient to maintain the $\mathbf{R}$ matrix throughout the algorithm.

4.2. SRIF Predict

Performing prediction in the SRIF formulation is significantly more complex than by using the traditional EKF formulation. The main results are presented here; for details please refer [5, Chapter 6]. The predicted value of $\mathbf{R}$ at time t_{k+1}, i.e. $\mathbf{R}_{k+1|k}$ is obtained by the following QR factorization:

$$QR\left(\begin{bmatrix} \mathbf{R}_w & \mathbf{0} \\ -\mathbf{R}_k^d\mathbf{G}_{eff} & \mathbf{R}_k^d \end{bmatrix}\right) = \begin{bmatrix} \tilde{\mathbf{R}}_w & \mathbf{R}_{wx} \\ \mathbf{0} & \mathbf{R}_{k+1|k} \end{bmatrix} \quad (26)$$

where, $\mathbf{Q}_d^{-1} = \mathbf{R}_w^T\mathbf{R}_w$, $\mathbf{R}_k^d = \mathbf{R}_{k|k}\Phi_k^{-1}$, and

$$\mathbf{G}_{eff} = \begin{bmatrix} \mathbf{G} \\ \mathbf{0} \end{bmatrix}, \Phi_k = \begin{bmatrix} \Phi_k^D & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \quad (27)$$

$\mathbf{Q}$ and Φ_k^D are as defined in subsection (3.2). At the face of it, it seems that a QR factorization of $(N_w + N_x) \times (N_w + N_x)$ sized matrix needs to be performed, where $N_x = 12 + 6N$. This is significantly more computationally intensive than the predict step of the conventional EKF. Observe, however, that $\mathbf{R}_{k|k}$ can be partitioned as

$$\mathbf{R}_{k|k} = \begin{bmatrix} \mathbf{R}_1 & \mathbf{R}_{12} \\ \mathbf{0} & \mathbf{R}_2 \end{bmatrix} \quad (28)$$

where, $\mathbf{R}_1$ is 12×12, and $\mathbf{R}_2$ is $6N \times 6N$. Hence,

$$\begin{aligned} \mathbf{R}_k^d &= \begin{bmatrix} \mathbf{R}_1 \left(\Phi_k^D\right)^{-1} & \mathbf{R}_{12} \\ \mathbf{0} & \mathbf{R}_2 \end{bmatrix}, \\ \mathbf{R}_k^d\mathbf{G}_{eff} &= \begin{bmatrix} \mathbf{R}_1 \left(\Phi_k^D\right)^{-1} \mathbf{G} \\ \mathbf{0} \end{bmatrix}, \end{aligned} \quad (29)$$

which, on substituting in Eq. (26) yields the following as the expression for the matrix to be factorized

$$\begin{bmatrix} \mathbf{R}_w & \mathbf{0} & \mathbf{0} \\ -\mathbf{R}_1 \left(\Phi_k^D\right)^{-1} \mathbf{G} & \mathbf{R}_1 \left(\Phi_k^D\right)^{-1} & \mathbf{R}_{12} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_2 \end{bmatrix} \quad (30)$$

Observe that the lower-right portion corresponding to $\mathbf{R}_2$ is already upper-triangular. Hence, triangularizing via QR needs to be performed only the upper $N_w + 12$ rows, which reduces the complexity of this step from $O(N^3)$ to $O(N^2)$, a significant reduction.

4.3. Feature update and removal

Addition of new features in the SRIF formulation is straightforward. Similar to feature addition in the EKF (subsection (3.4)), the state vector and the $\mathbf{R}$ matrix are augmented with the entries of the newly initialized values,

$$\mathbf{x}_{k^+} = \begin{bmatrix} \mathbf{x}_{k^-} \\ \mathbf{x}_{new} \end{bmatrix}, \mathbf{R}_{k^+} = \begin{bmatrix} \mathbf{R}_{k^-} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{new} \end{bmatrix} \quad (31)$$

Seq#	EKF single precision		EKF fixed point (24.40)		SRIF single precision		SRIF fixed-point (16.16)	
	TranError [%]	RotError (deg/m)	TranError [%]	RotError (deg/m)	TranError [%]	RotError (deg/m)	TranError [%]	RotError (deg/m)
1	0.87	9.40E-05	0.88	2.08E-06	0.87	2.50E-04	1.95	1.77E-03
2	8	5.50E-05	10.22	5.47E-07	9.83	8.06E-05	11.24	6.29E-04
3	1.68	1.04E-04	1.67	1.75E-06	1.68	1.31E-04	3.66	2.07E-03
4	1.83	6.64E-05	1.77	1.54E-06	1.83	7.40E-05	3.18	3.57E-03
5	0.67	5.79E-05	0.67	4.90E-07	0.67	9.53E-05	6.96	1.59E-03
6	1.99	1.47E-04	1.51	1.86E-06	1.99	1.36E-04	2.59	2.16E-03
7	4.9	1.74E-04	5.31	1.24E-06	5	2.75E-04	7.69	2.05E-03
8	1.67	1.28E-04	1.69	2.13E-06	1.67	1.77E-04	3.33	3.95E-03
9	1.42	1.39E-04	1.54	1.29E-06	1.51	8.59E-05	4.16	4.24E-03
10	1.82	1.41E-04	1.88	6.70E-07	1.82	7.55E-05	3.6	1.99E-03

Table 1. EKF and SRIF results with various precisions.

Note that $\mathbf{R}_{new}$ is upper triangular, and hence so is $\mathbf{R}_k$.

Removal of features is more complex than in the EKF case and cannot be accomplished by a simple deletion of rows and columns. A two-step process is followed. First, a permutation matrix Π is applied to the state-vector $\mathbf{x}$ (subscript k dropped for notational convenience) such that it is partitioned into two sub-vectors: sub-vector $\mathbf{x}_d$ of length d representing the features that need to be discarded and sub-vector $\mathbf{x}_r$ of length r corresponding to features that need to be retained, i.e. $\mathbf{x}_\Pi = \mathbf{\Pi}x = \begin{bmatrix}\mathbf{x}_d^T \mathbf{x}_r^T\end{bmatrix}^T$. Then, the covariance of $\mathbf{x}_\Pi$ is $\mathbf{P}_\Pi = \mathbf{\Pi P \Pi}^T$. Hence, $\mathbf{P}_\Pi^{-1} = \mathbf{\Pi P}^{-1}\mathbf{\Pi}^T$, since $\mathbf{\Pi}^{-1} = \mathbf{\Pi}^T$ for any permutation matrix. Now, $\mathbf{P}^{-1} = \mathbf{R}^T\mathbf{R}$. Hence, $\mathbf{P}_\Pi^{-1} = \mathbf{R}_\Pi^T\mathbf{R}_\Pi$, where $\mathbf{R}_\Pi = \mathbf{R}\mathbf{\Pi}^T$. Clearly, $\mathbf{R}_\Pi$ is not upper triangular, but is easily triangularized by a applying a sequence of Givens rotations. Now, suppose that $\mathbf{R}_\Pi$ is partitioned after triangularization as

$$\mathbf{R}_\Pi = \begin{bmatrix}\mathbf{R}_d & \mathbf{R}_{dr} \\ \mathbf{0} & \mathbf{R}_r\end{bmatrix} \tag{32}$$

Then,

$$\mathbf{R}_\Pi^{-1} = \begin{bmatrix}\bar{\mathbf{R}}_d & \bar{\mathbf{R}}_{dr} \\ \mathbf{0} & \bar{\mathbf{R}}_r\end{bmatrix} = \begin{bmatrix}\mathbf{R}_d^{-1} & -\mathbf{R}_d^{-1}\mathbf{R}_{dr}\mathbf{R}_r^{-1} \\ \mathbf{0} & \mathbf{R}_r^{-1}\end{bmatrix} \tag{33}$$

Hence, $\mathbf{P}_\Pi$ can be expressed as

$$\begin{aligned}\mathbf{P}_\Pi \triangleq \begin{bmatrix}\mathbf{P}_d & \mathbf{P}_{dr} \\ \mathbf{P}_{dr}^T & \mathbf{P}_r\end{bmatrix} &= \mathbf{R}_\Pi^{-1}\mathbf{R}_\Pi^{-T} \\ &= \begin{bmatrix}\bar{\mathbf{R}_d}\bar{\mathbf{R}_d}^T + \bar{\mathbf{R}_r}\bar{\mathbf{R}_r}^T & \bar{\mathbf{R}_{dr}}\bar{\mathbf{R}_r}^T \\ \bar{\mathbf{R}_r}\bar{\mathbf{R}_{dr}}^T & \bar{\mathbf{R}_r}\bar{\mathbf{R}_r}^T\end{bmatrix}\end{aligned} \tag{34}$$

From the above, it is seen that the covariance matrix, $\mathbf{P}_r$ of the retained features is given by $\mathbf{P}_r = \bar{\mathbf{R}_r}\bar{\mathbf{R}_r}^T$ and hence $\mathbf{P}_r^{-1} = \mathbf{R}_r^T\mathbf{R}_r$. Hence, the R-matrix of the remaining features is lower-right $r \times r$ submatrix of the triangularized $\mathbf{R}_\Pi$.

5. Experiments and Results

In order to demonstrate the numerical superiority of the SRIF-based approach, both EKF-SfM and SRIF-SfM were run on a set of test clips with numerical data-types of various precisions. The objective here is to evaluate the degradation in accuracy occurring due to the use of lower-precision numerical-types rather than to evaluate the inherent quality of the SfM algorithm itself. The test-set includes ten clips from the Kitti Odometry dataset [12] for which ground truth poses are available. Details about the test clips are provided in the supplementary material. To evaluate and compare accuracy, the methodology proposed in [12] is adopted wherein the relative translational and rotational errors are calculated over fixed-interval segments of length 100, and the average error over these segments is reported in Table 1. Only those numerical types that are commonly supported in *hardware* on modern processors were tested. Non-standard types (such as 40-bit integer, 48-bit integer, etc.) were not considered because these would involve implementing bit-shift operations for various arithmetic operators *in software* which would result in a significant performance degradation.

For EKF-SfM, the results are reported with single-precision (32-bit) floating-point and with 64-bit fixed-point in Q23.40 format. Reducing the precision resulted in the covariance matrix, $\mathbf{P}$, losing positive definiteness, which also resulted in $\mathbf{S}$ becoming indefinite. This caused the Cholesky factorization of $\mathbf{S}$ to fail. Using LDL factorization for $\mathbf{S}$ in lieu of Cholesky did not help because although $\mathbf{S}$ would factorize with LDL, the outputs using an indefinite $\mathbf{S}$ quickly diverged and became meaningless.

For SRIF-SfM, results are reported for single-precision floating-point and with 32-bit fixed-point in Q15.16 format. It was observed that with 32-bit integer arithmetic, the translational error degraded only slightly compared to floating-point. Rotational error could be up-to an order of magnitude

worse; however, in absolute terms it was still very small. From this it is clear that the SRIF-based SfM is capable at operating with roughly half the number of bits as EKF-based SfM.

6. Computational Complexity

The computational complexity of the SRIF-based SfM is analyzed and compared to that of the EKF-based SfM and the results are presented in Table 2. Complexity figures are reported only for the steps that require a significant compute; steps for which the computational cost is negligible compared to the overall cost are ignored. Furthermore, for any given step, only the highest order terms are reported as these dominate the cost. The time subscripts on various matrices are dropped for notational convenience. The special structures of matrices involved are leveraged to reduce complexity when applicable (such as sparsity of $\mathbf{H}$, the approximately upper-triangular form of $\mathbf{R}_{\Pi}$, etc.). For a more general comparison of the computational complexity of square-root filtering vs Kalman filtering, please refer [5]. The derivations to arrive at the figures presented in Table 2 are straightforward but too lengthy to include here, and are hence included in the supplementary material.

For the feature removal-step in SRIF-SfM, it is not possible to arrive at a closed-from expression for its complexity because it does not simply depend on the number of features being removed, but also on which particular features are being removed. It is, however, possible to demonstrate that this term has an upper-bound of $162N^3$. This upper-bound is attained in a highly specific scenario (explained in the supplementary material) which is rarely, if ever, realized. Typically, the cost of the feature-removal step is a small fraction, β, of this upper-bound. From our experiments over several hundreds of input frames, the average value of β was observed to be 0.084 and the maximum value observed was 0.235.

6.1. Implementation on processors

From Table 2, it can be seen that SRIF-SfM typically requires fewer operations than the EKF, which already indicates that the real-world performance of the SRIF-based SfM should be faster than that of the EKF-based one. More importantly, though, SRIF-SfM can be implemented with 32-bit integer type instead of the 64-bit integer or single-precision floating-point type required for EKF. Throughput of 32-bit integer operations is twice that of 64-bit integer operations. Hence, the performance of SRIF (as measured by operations/sec) will be 2x that of EKF when comparing implementations using fixed-point arithmetic.

Comparing performance of SRIF-SfM using 32-bit integer arithmetic to that of the EKF-SfM using single-precision (32-bit) floating-point is less straightforward. Fixed-point architectures, by definition, do not provide capabilities for automatic adjustment of the exponent. Floating-point architecture, on the other hand, must be capable of adjusting the exponent or normalizing-factor automatically. This increases the hardware cost and the power consumption. In order to balance cost versus throughput, the fixed-point embedded processors are typically twice as fast as floating-point processors [3] and are half the price of the floating-point processors. The use of floating point processors, therefore, significantly increases the system cost along with the power consumption. The recent trend is to use FPGAs for the special purpose applications in order to have rapid time-to-market along with low-power real-time implementations. If there is no special hardware block for floating point operations, the fixed-point versus the floating-point multiplier on an FPGA results in four times increase in size and latency[25], which is effectively a 16x performance difference. In the presence of specialized hardware support, the floating-point implementation on an FPGA will take double the area, along with the increase in platform cost and power consumption [19].

In short, use of floating-point arithmetic will require paying a price either in terms of performance, or power, or cost and algorithms that are able to run using fixed-point arithmetic will generally be easier to implement such on constrained platforms. In fact, the use of SRIF might enable implementation on low-power embedded platforms that support only lower-precision types owing to which they are unable to support a traditional EKF implementation.

7. Conclusions and Future Work

In this work, a SRIF-based approach to the real-time monocular structure-from-motion problem is presented. It is demonstrated that our SfM algorithm can be implemented using 32-bit integer arithmetic, the first such implementation to the best of our knowledge. This should enable implementation of SfM on low-power processors that either do not support floating-point arithmetic, or if they do, do so at tremendous cost to performance or power. The ability to work with lower-precision arithmetic is a direct consequence of using the square-root information filter instead of the traditional EKF. Going forward, therefore, we expect to implement real-time stereo-SfM and RGB-D SfM using SRIF formulations too in the anticipation that these will be able to run with even lower-precision arithmetic.

References

[1] A. Azarbayejani and A. P. Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):562–575, 1995. 1

[2] B. M. Bell and F. W. Cathey. The iterated kalman filter update as a gauss-newton method. *IEEE Transactions on Automatic Control*, 38(2):294–297, 1993. 5

	EKF		SRIF	
	Step	Complexity	Step	Complexity
Update				
	$\mathbf{PH}^T$	$288N^2$	$\mathbf{A} = \mathbf{Q}\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix}$	$144N^3$
	$\mathbf{S} = \mathbf{HPH}^T + \mathbf{Q}_v$	$100N^2$	$\mathbf{Q}^T\Delta\mathbf{e}_{k+1}$	$120N^2$
	$\mathbf{S} = \mathbf{R}_s^T\mathbf{R}_s$	$8N^3/3$	$\mathbf{R}^{-1}\mathbf{Q}^T\Delta\mathbf{e}_{k+1}$	$36N^2$
	$\mathbf{K} = \mathbf{PH}^T\mathbf{R}_s^{-1}\mathbf{R}_s^{-T}$	$48N^3$		
	$\mathbf{P} = \mathbf{P} - \mathbf{KHP}$	$144N^2$		
Predict				
		Negligible		Negligible
Feature removal				
		-	$\mathrm{QR}(\mathbf{R}_\Pi)$	$\beta \cdot 162N^3$
Total				
		$194.67N^3$		$144N^3 + \beta \cdot 162N^3$ $= 157.61N^3, \beta = 0.084$

Table 2. Computational complexity of EKF and SRIF

[3] I. Berkeley Design Technology. BDTI DSP Kernel Benchmarks (BDTImark2000) Certified Results, 2016. 7

[4] G. J. Bierman. Sequential square root filtering and smoothing of discrete linear systems. *Automatica*, 10(2):147–158, 1974. 1, 5

[5] G. J. Bierman. *Factorization methods for discrete sequential estimation*, volume 128 of *Mathematics in Science and Engineering*. New York: Academic, 1977. 5, 7

[6] J. Civera, A. J. Davison, and J. M. Montiel. Inverse depth parametrization for monocular slam. *IEEE transactions on Robotics*, 24(5):932–945, 2008. 1, 3, 4

[7] L. A. Clemente, A. J. Davison, I. D. Reid, J. Neira, and J. D. Tards. Mapping large loops with a single hand-held camera. *Robotics: Science and Systems*, 2(2):2, 2007. 1

[8] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007. 1

[9] F. Dellaert and M. Kaess. Square root sam: Simultaneous location and mapping via square root information smoothing. *Int. Journal of Robotics Research*, pages 1181–1203, 2006. 2

[10] M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001. 1

[11] C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014. 1

[12] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 6

[13] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50, 1988. 4

[14] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *Int. Journal of Robotics Research*, 31(2):216–235, 2011. 2

[15] M. Kaess, A. Ranganathan, and F. Dellaert. isam: Fast incremental smoothing and mapping with efficient data association. In *ICRA*, pages 1670–1677, 2007. 2

[16] P. G. Kaminski, A. Bryson, and S. . Schmidt. Discrete square root filtering: A survey of current techniques. *IEEE Trans. Automat. Contr.*, pages 727–736, 1971. 5

[17] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. *ISMAR*, pages 225–234, 2007. 1

[18] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. *ISMAR*, pages 83–86, 2009. 1

[19] M. Langhammer and B. Pasca. Design and implementation of an embedded FPGA floating point dsp block. In *IEEE 22nd Symposium on Computer Arithmetic (ARITH*. 7

[20] C. F. V. Loan. Computing integrals involving the matrix exponential. *IEEE Trans. Automat. Control*, AC-23:395–404, 1978. 4

[21] S. Middelberg, T. Sattler, O. Untzelmann, and L. Kobbelt. Scalable 6-dof localization on mobile devices. *ECCV*, pages 268–283, 2014. 1

[22] A. I. Mourikis and S. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Int. Conf. Robotics and Automation*, pages 3565–3572. IEEE, 2007. 4

[23] J. E. . Potter and R. G. . Stem. Statistical filtering of space navigation measurements. In *AIAA Guidance Contr. Conf.*, 1963. 5

[24] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *International con-*

ference on computer vision, pages 2564–2571. IEEE, 2011. 4

[25] C. Te Ewe, P. Y. Cheung, and G. A. Constantinides. Dual fixed-point: An efficient alternative to floating-point computation. In *International Conference on Field Programmable Logic and Applications*, pages 200–208. Springer, 2004. 7

[26] J. Ventura, C. Arth, G. Reitmayr, and D. Schmalstieg. Global localization from monocular slam on a mobile phone. *IEEE transactions on visualization and computer graphics*, 20(4):531–539, 2014. 1

[27] M. Verhaegen and P. V. Dooren. Numerical aspects of different kalman filter implementations. *IEEE Trans. Automat. Contr.*, pages 907–917, 1986. 5

[28] B. Williams and I. Reid. On combining visual slam and visual odometry. *ICRA*, pages 3494–3500, 2010. 1