Complexity Analysis: A Linear-Complexity EKF for Visual-Inertial Navigation with Loop Closures

Patrick Geneva - pgeneva@udel.edu
Kevin Eckenhoff - keck@udel.edu
Guoquan Huang - ghuang@udel.edu

Department of Mechanical Engineering
University of Delaware, Delaware, USA

RPNG

Robot Perception and Navigation Group (RPNG)
Tech Report - RPNG-2019-LOOP
Last Updated - February 8, 2019

Contents

1 State Propagation

1.1 Problem Formulation

During propagation we process a set inertial measurements to move the state mean and covariance forward in time. To propagate the state covariance matrix forward, we use the discrete time state transition matrix as follows:

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{a}_{m_k} - \mathbf{n}_{a_k}, \boldsymbol{\omega}_{m_k} - \mathbf{n}_{\omega_k}) \tag{1}$$

$$\mathbf{P}_{k|k-1} = \begin{bmatrix} \boldsymbol{\Phi}_{k-1}\mathbf{P}_{AA_{k-1|k-1}}\boldsymbol{\Phi}_{k-1}^\top & \boldsymbol{\Phi}_{k-1}\mathbf{P}_{AS_{k-1|k-1}} \\ \mathbf{P}_{SA_{k-1|k-1}}\boldsymbol{\Phi}_{k-1}^\top & \mathbf{P}_{SS_{k-1|k-1}} \end{bmatrix} + \begin{bmatrix} \mathbf{Q}_{k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{2}$$

1.2 Complexity Analysis

We consider that the current active state variables $\mathbf{x}_{A_{k|k}}$ to have an error state of size a and the Schmidt state variables $\mathbf{x}_{S_{k|k}}$ to have an error state of size n. The mean propagation only affects the active state variables, thus we only focus on the state covariance propagation. We have the following algorithm and computational costs for a set of IMU measurements $\mathcal{I}$ that are of some size q:

Algorithm 1 Schmidt-MSCKF Covariance Propagation

		cost	times
1:	**procedure** COVARIANCE_PROPAGATION($\mathbf{P}_{AA_{k-1\|k-1}}$, $\mathbf{P}_{AS_{k-1\|k-1}}$, $\mathcal{I}$)		
2:	// Set initial values for state transition and noise		
3:	$\boldsymbol{\Phi} = \mathbf{I}_{a\times a}$	a^2	1
4:	$\mathbf{Q} = \mathbf{0}_{a\times a}$	a^2	1
5:	// Compound all inertial measurements		
6:	**for** $\omega_i, \mathbf{a}_i \in \mathcal{I}$ **do**	1	q
7:	$\boldsymbol{\Phi} = \boldsymbol{\Phi}(i+1, i)\boldsymbol{\Phi}$	$a^3 + a^2$	q
8:	$\mathbf{Q} = \boldsymbol{\Phi}(i+1, i)\mathbf{Q}\boldsymbol{\Phi}(i+1, i)^\top + \mathbf{Q}_i$	$2a^3 + 2a^2$	q
9:	**end for**		
10:	// Update active covariance, and Schmidt cross-terms		
11:	$\mathbf{P}_{AA_{k\|k-1}} = \boldsymbol{\Phi}\mathbf{P}_{AA_{k-1\|k-1}}\boldsymbol{\Phi}^\top + \mathbf{Q}$	$2a^3 + 2a^2$	1
12:	$\mathbf{P}_{AS_{k\|k-1}} = \boldsymbol{\Phi}\mathbf{P}_{AS_{k-1\|k-1}}$	$a^2n + an$	1
13:	**end procedure**		

It is clear to see that the most expensive computation is the final multiplication of the Schmidt cross-term covariance. We have the following computational cost for propagation if we take the size of the active state to remain constant over time:

$$\boxed{\text{mean propagation} : O(1)} \tag{3}$$

$$\boxed{\text{covariance propagation} : O(n)} \tag{4}$$

2 Clone Marginalization

2.1 Problem Formulation

When a clone leaves the sliding window, we chose if we should add that clone to the Schmidt state, or marginalize it out. Here we look at what happens when we marginalize a clone from the end of our state.

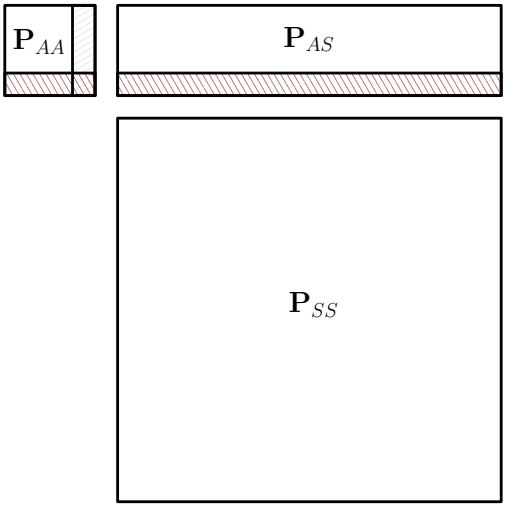

Figure 1: Illustrate of what is deleted upon marginalization of a clone. The rows shown in red will be deleted after the process is finished.

2.2 Complexity Analysis

We consider that the current active state variables $\mathbf{x}_{A_{k|k}}$ to have an error state of size a and the Schmidt state variables $\mathbf{x}_{S_{k|k}}$ to have an error state of size n. We assume that the clone that will be marginalized is at the *end* of the active variable state/covariance.

Algorithm 2 Schmidt-MSCKF Covariance Clone Marginalization

		cost	times
1:	**procedure** COVARIANCE_CLONE_MARG($\mathbf{P}_{AA}$, $\mathbf{P}_{AS}$, $\mathbf{P}_{SS}$)		
2:	$\mathbf{P}_{AA}$.resize($P_{AA}.r-6, P_{AA}.c-6$)	c_1	1
3:	$\mathbf{P}_{AS}$.resize($P_{AS}.r-6, P_{AS}.c$)	c_2	1
4:	**end procedure**		

From the above, we can see that the time to resize the matrices should dominate the actual cost. In practice we don't deallocate memory and simply keep track of the current size of the covariance, thus we don't have a cost to resize the matrix allowing for constant computation cost. Considering the active state remains constant over time we have the following costs:

$$\boxed{\text{mean clone marginalization} : O(1)} \tag{5}$$

$$\boxed{\text{covariance clone marginalization} : O(1)} \tag{6}$$

3 Keyframe Augmentation

3.1 Problem Formulation

When a clone leaves the sliding window, we chose if we should add that clone to the Schmidt state, or marginalize it out. If we are going to add it to the Schmidt state, we call this operation "keyframe augmentation" as we are adding a new keyframe into our Schmidt state.

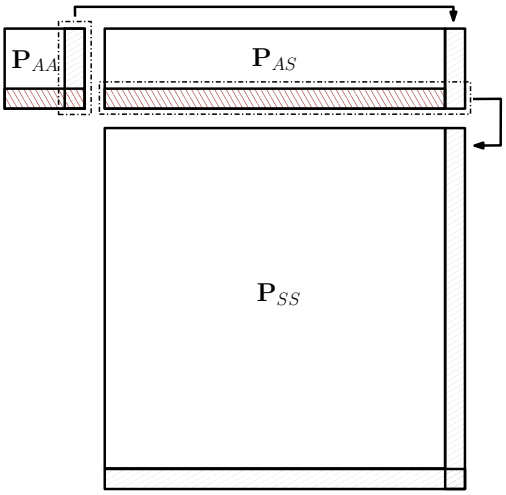

Figure 2: Illustrate of what is added and deleted upon keyframe augmentation. The rows shown in red will be deleted after the process is finished, while the rows shown in green have been added by copying the cross-terms from the $\mathbf{P}_{AA}$ and $\mathbf{P}_{AS}$ matrices.

3.2 Complexity Analysis

We consider that the current active state variables $\mathbf{x}_{A_{k|k}}$ to have an error state of size a and the Schmidt state variables $\mathbf{x}_{S_{k|k}}$ to have an error state of size n. For the mean, we simply need to append the new 6×1 to the end our Schmidt state vector, and remove it from our active state vector. Thus, we focus on how to re-order the covariance such that it has the new Schmidt variable at the end of it. We assume that the clone that will be moved to the Schmidt state is at the *end* of the active variable state/covariance.

Algorithm 3 Schmidt-MSCKF Covariance Keyframe Augmentation

		cost	times
1:	**procedure** COVARIANCE_KEY_AUG($\mathbf{P}_{AA}$, $\mathbf{P}_{AS}$, $\mathbf{P}_{SS}$)		
2:	// Resize our covariance matrix		
3:	$\mathbf{P}_{AS}$.resize($P_{AS}.r, P_{AS}.c$+6)	c_1	1
4:	$\mathbf{P}_{SS}$.resize($P_{SS}.r$+6, $P_{SS}.c$+6)	c_2	1
5:	// Copy from active state to cross-terms		
6:	$\mathbf{P}_{AS}(0, P_{AS}.c-6, P_{AS}.r, 6)$=$\mathbf{P}_{AA}(0, P_{AA}.c-6, P_{AA}.r, 6)$	$6a$	1
7:	// Copy from cross-terms to Schmidt		
8:	$\mathbf{P}_{SS}(P_{SS}.r-6, 0, 6, P_{SS}.c)$=$\mathbf{P}_{AS}(P_{AS}.r-6, 0, 6, P_{AS}.c)$	$6n$	1
9:	$\mathbf{P}_{SS}(0, P_{SS}.c-6, P_{SS}.r, 6)$=$\mathbf{P}_{AS}(P_{AS}.r-6, 0, 6, P_{AS}.c)^\top$	$6n$	1
10:	// Finally, reduce size of covariances		
11:	$\mathbf{P}_{AA}$.resize($P_{AA}.r-6, P_{AA}.c-6$)	c_3	1
12:	$\mathbf{P}_{AS}$.resize($P_{AS}.r-6, P_{AS}.c$)	c_4	1
13:	**end procedure**		

From the above, we can see that the time to resize the matrices should dominate the actual cost of copying the covariance elements. In practice we preallocate memory and simply keep track of the current size of the covariance, thus we don't have a cost to resize the matrix. Considering the active state remains constant over time we have the following costs:

$$\boxed{\text{mean augmentation} : O(1)} \tag{7}$$

$$\boxed{\text{covariance augmentation} : O(n)} \tag{8}$$

4 EKF Update

4.1 Problem Formulation

Having propagated the state, we can update the state estimate means and covariance as follows:

$$\hat{\mathbf{x}}_{A_{k|k}} = \hat{\mathbf{x}}_{A_{k|k-1}} + \mathbf{K}_{A_k}\tilde{\mathbf{z}}'_k \tag{9}$$

$$\hat{\mathbf{x}}_{S_{k|k}} = \hat{\mathbf{x}}_{S_{k|k-1}} \tag{10}$$

With the Schmidt Kalman gain $\mathbf{K}_{S_k} = \mathbf{0}$, we immediately have the covariance update as follows:

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \begin{bmatrix} \mathbf{K}_{A_k}\mathbf{S}_k\mathbf{K}_{A_k}^\top & \mathbf{K}_{A_k}\mathbf{H}'_k\begin{bmatrix}\mathbf{P}_{AS_{k|k-1}}\\ \mathbf{P}_{SS_{k|k-1}}\end{bmatrix} \\ \begin{bmatrix}\mathbf{P}_{AS_{k|k-1}}\\ \mathbf{P}_{SS_{k|k-1}}\end{bmatrix}^\top \mathbf{H}'^\top_k\mathbf{K}_{A_k}^\top & \mathbf{0} \end{bmatrix} \tag{11}$$

$$= \mathbf{P}_{k|k-1} - \begin{bmatrix} \mathbf{L}_{A_k}\mathbf{S}_k^{-1}\mathbf{L}_{A_k}^\top & \mathbf{L}_{A_k}\mathbf{S}_k^{-1}\mathbf{L}_{S_k}^\top \\ \mathbf{L}_{S_k}\mathbf{S}_k^{-1}\mathbf{L}_{A_k}^\top & \mathbf{0} \end{bmatrix} \tag{12}$$

where the standard Kalman gain can be defined as:

$$\begin{aligned}\mathbf{K}_k = \begin{bmatrix}\mathbf{K}_{A_k}\\ \mathbf{K}_{S_k}\end{bmatrix} &= \begin{bmatrix}\mathbf{P}_{AA_{k|k-1}}\mathbf{H}_{A_k}^\top + \mathbf{P}_{AS_{k|k-1}}\mathbf{H}_{S_k}^\top \\ \mathbf{P}_{SA_{k|k-1}}\mathbf{H}_{A_k}^\top + \mathbf{P}_{SS_{k|k-1}}\mathbf{H}_{S_k}^\top\end{bmatrix}\mathbf{S}_k^{-1} \\ &=: \begin{bmatrix}\mathbf{L}_{A_k}\\ \mathbf{L}_{S_k}\end{bmatrix}\mathbf{S}_k^{-1}\end{aligned} \tag{13}$$

where:

$$\mathbf{S}_k = \mathbf{H}'_k\mathbf{P}_{k|k-1}\mathbf{H}'^\top_k + \mathbf{R}'_k \tag{14}$$

4.2 Complexity Analysis

We consider the current active state variables $\mathbf{x}_{A_{k|k}}$ to have an error state of size a, the Schmidt state variables $\mathbf{x}_{S_{k|k}}$ to have an error state of size n, and the size of the measurement residuals to be of size q. We define the set of variables that the update involves as $\mathcal{K}$ which is sparse and has $|\mathcal{K}|$ non-zero elements (i.e. $\mathcal{K} \ll n$).

For the state mean during update, once the Kalman gain $\mathbf{K}_{A_k}$ is computed, it can be clearly seen that it is only of complexity of the active state and the number of measurements contained in $\tilde{\mathbf{z}}'_k$. This Kalman gain $\mathbf{K}_{A_k}$ will be found through the computation of the updated covariance, and is the most expensive computation during the mean update.

Algorithm 4 Schmidt-MSCKF Covariance Update

		cost	times
1:	**procedure** COVARIANCE_UPDATE($\mathbf{P}_{AA}$, $\mathbf{P}_{AS}$, $\mathbf{P}_{SS}$, $\mathbf{H}_k$)		
2:	// First, lets compute the sub-matrices $\mathbf{L}_{A_k}$ and $\mathbf{L}_{S_k}$		
3:	$\mathbf{L}_{A_k} = \mathbf{0}_{a \times q}$	aq	1
4:	$\mathbf{L}_{S_k} = \mathbf{0}_{n \times q}$	nq	1
5:	// Now we loop through all the sparse $\mathbf{H}_k$ for each state variable scalar (whose *id* is its location in the covariance matrix)		
6:	**for all** active state variables **do**	1	a
7:	$\mathbf{L}_{A_k}(id, 0, 1, q) = \sum_{k \in \mathcal{K}} P_{ik} \mathbf{H}_k^\top$	$2\|\mathcal{K}\|q$	a
8:	**end for**		
9:	**for all** schmidt state variables **do**	1	n
10:	$\mathbf{L}_{S_k}(id, 0, 1, q) = \sum_{k \in \mathcal{K}} P_{ik} \mathbf{H}_k^\top$	$2\|\mathcal{K}\|q$	n
11:	**end for**		
12:	// Compute the smaller covariance matrix by *element*, only involving the variables that the Jacobian $\mathbf{H}_k$ involves.		
13:	$\mathbf{HPH}^\top = \sum_{(i,j) \in \mathcal{K} \times \mathcal{K}} \mathbf{H}_i P_{ij} \mathbf{H}_j^\top$	$2\|\mathcal{K}\|^2 q^2$	1
14:	// Compute $\mathbf{S}_k$ and its inverse		
15:	$\mathbf{S}_k = \mathbf{HPH}^\top + \mathbf{R}'_k$	$2q^2$	1
16:	$\mathbf{S}_k^{-1} = \text{inv}(\mathbf{S}_k)$	$q^3 + q^2$	1
17:	// Finally, do the covariance update		
18:	$\mathbf{P}_{AA} = \mathbf{P}_{AA} - \mathbf{L}_{A_k} \mathbf{S}_k^{-1} \mathbf{L}_{A_k}^\top$	$aq^2 + a^2 q + 2a^2$	1
19:	$\mathbf{P}_{AS} = \mathbf{P}_{AS} - \mathbf{L}_{A_k} \mathbf{S}_k^{-1} \mathbf{L}_{S_k}^\top$	$aq^2 + aqn + 2an$	1
20:	**end procedure**		

It is important to note that the size non-zero entries in the sparse Jacobian $\mathbf{H}_k$ which involves $\mathcal{K}$ elements (which we have defined as size $|\mathcal{K}|$), directly impacts the complexity of the update. To ensure that update is linear in time, we enforce that at each clone time (i.e., image time) we only matche to a single keyframe. This ensures that the maximum keyframes that we will ever match to is the size of the sliding window (i.e., no more then a in size). The impact this design decision has on the number of non-zero elements in $\mathbf{H}_k$ is that the it is never more then the number of state elements, thus we can consider it constant in terms of big-O. Thus, we have the following results:

$$\boxed{\text{mean update} : O(n)} \tag{15}$$

$$\boxed{\text{covariance update} : O(n)} \tag{16}$$