The Battle for Filter Supremacy: A Retrospective: The Movie: The Game: The Paper

Lee Clement[1] and Valentin Peretroukhin[1]

***Abstract*— Monkey Style Chinese Kung Fu**

I. INTRODUCTION

The combination of visual and inertial sensors is a powerful tool for autonomous navigation in unknown environments. Indeed, cameras and inertial measurement units (IMUs) are complementary in several respects. Since an IMU directly measures accelerations and rotational velocities, these values must be integrated to arrive at a new pose estimate. However, the noise inherent in the IMU's measurements is included in the integration as well, and consequently the pose estimates can drift unbounded over time. The addition of a camera is an excellent way to bound this cumulative drift error since the camera's signal-to-noise ratio is highest when the camera is moving slowly. On the other hand, cameras are not robust to motion blur induced by large accelerations. In these cases, the strength of the IMU's signal far exceeds its baseline noise and can be relied upon more heavily in estimating pose changes.

The question, then, is how best to fuse measurements from these two sensor types to arrive at an accurate estimate of a vehicle's motion over time. A complicating factor in the general form of this problem is the absence of a known map of features from which the camera can generate measurements. Any solution must therefore solve a Simultaneous Localization and Mapping (SLAM) problem, although the importance placed on the mapping component may vary from algorithm to algorithm.

In this work, we compare and contrast three modern solutions, the Extended Kalman Filter (EKF), the Sliding Window Filter (SWF), and a hybrid solution, the Multi-State Constraint Kalman Filter (MSCKF), that combines the strengths of both. We present an experimental comparison of the SWF and MSCKF using a dataset consisting of IMU and camera data with accurate ground truth for both sensor motion and landmark positions.

A. Extended Kalman Filter (EKF)

In the Extended Kalman Filter (EKF) solution, vehicle poses and feature positions are simultaneously estimated at each time step by augmenting the filter state with feature positions. This technique, sometimes referred to as EKF-SLAM, attempts to track pose changes and create a globally consistent map of features by recursively updating the state as new measurements become available.

[1]Institute for Aerospace Studies, University of Toronto, Toronto, ON, Canada {lee.clement, valentin.peretroukhin}@robotics.utias.utoronto.ca

Although the recursive nature of EKF-SLAM allows it to operate online, the computational cost of the filter grows cubically with map size. This behavior is due to the fact that the dimension of the state grows linearly with the number of features, and the computational cost of inverting the state covariance matrix while computing the Kalman gain is cubic in the dimension of the state. Consequently, the spatial extent over which EKF-SLAM can be used online is limited by the necessarily finite compute envelope available to it.

Another limitation of EKF-SLAM is that it is forgetful. Because the filter state includes only the most recent vehicle pose, a given update step can never modify past poses even if later feature measurements ought to constrain them. By locking in past poses, the EKF-SLAM formulation condemns itself to sub-optimally estimating both vehicle motion and feature positions.

B. Sliding Window Filter (SWF)

In contrast to EKF-SLAM, the aim of the Sliding Window Filter (SWF) is not to construct a globally consistent map, but rather to estimate a vehicle's motion by optimizing a sliding window of vehicle poses and feature positions. The optimization problem in the SWF is typically solved as a non-linear least squares problem using Gauss-Newton optimization or some other algorithm.

$$(\mathbf{H}^T\mathbf{T}^{-1}\mathbf{H})\delta\mathbf{x}^* = -\mathbf{H}^T\mathbf{T}^{-1}\mathbf{e}(\bar{\mathbf{x}}) \tag{1}$$

$$\delta\mathbf{x} := \begin{bmatrix}\delta\mathbf{x}_0 & \dots & \delta\mathbf{x}_K & \delta\mathbf{f}_1 & \dots & \delta\mathbf{f}_N\end{bmatrix}^T, \tag{2}$$

where N is the number of unique features observed in the window. The matrix $\mathbf{H}$ is given by (showing non-zero blocks only)

$$\mathbf{H} = \left[\begin{array}{cccccc}
-\mathbf{H}_{\mathbf{x},1} & \mathbf{1} & & & & \\
 & -\mathbf{G}_{\mathbf{x},1} & & & & -\mathbf{G}_{\mathbf{f},1} \\ \hline
 & -\mathbf{H}_{\mathbf{x},2} & \mathbf{1} & & & \\
 & & -\mathbf{G}_{\mathbf{x},2} & & & -\mathbf{G}_{\mathbf{f},2} \\ \hline
 & & \ddots & \ddots & & \vdots \\ \hline
 & & & -\mathbf{H}_{\mathbf{x},K} & \mathbf{1} & \\
 & & & & -\mathbf{G}_{\mathbf{x},K} & -\mathbf{G}_{\mathbf{f},K}
\end{array}\right], \tag{3}$$

where

$$\mathbf{G}_{\mathbf{x},k}^{j} = \frac{\partial \mathbf{g}}{\partial \mathbf{p}}\Big|_{\bar{\mathbf{p}}_{C_k}^{f_j C_k}} \begin{bmatrix} -\mathbf{C}_{CG}\mathbf{C}_{IG,k} \\ \mathbf{C}_{CG}(\mathbf{C}_{IG,k}(\boldsymbol{\rho}_i^{f_j G} - \mathbf{r}_G^{IG,k}))^{\times} \end{bmatrix}^T \tag{4}$$

$$\mathbf{G}_{\mathbf{x},k} = \begin{bmatrix} \mathbf{G}_{\mathbf{x},k}^{1} & \dots & \mathbf{G}_{\mathbf{x},k}^{M} \end{bmatrix}^T \tag{5}$$

$$\mathbf{G}_{\mathbf{f},k}^{j} = \frac{\partial \mathbf{g}}{\partial \mathbf{p}}\Big|_{\bar{\mathbf{p}}_{C_k}^{f_j C_k}} \mathbf{C}_{CI}\mathbf{C}_{IG,k} \tag{6}$$

$$\mathbf{G}_{\mathbf{f},k} = \underbrace{\begin{bmatrix} \mathbf{G}_{\mathbf{f},k}^{1} & & & \\ & \mathbf{G}_{\mathbf{f},k}^{2} & & \\ & \vdots & & \\ & & & \mathbf{G}_{\mathbf{f},k}^{M} \end{bmatrix}}_{j\text{th block column position given by feature ID } j} \tag{7}$$

with

$$\bar{\mathbf{p}}_{C_k}^{f_j C_k} := \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{C}_{CI}\left(\mathbf{C}_{IG,k}(\boldsymbol{\rho}_G^{f_j G} - \mathbf{r}_G^{IG,k}) - \boldsymbol{\rho}_I^{CI}\right), \tag{8}$$

$$\frac{\partial \mathbf{g}}{\partial \mathbf{p}}\Big|_{\bar{\mathbf{p}}_{C_k}^{f_j C_k}} = \frac{1}{z^2}\begin{bmatrix} f_u z & 0 & -f_u x \\ 0 & f_v z & -f_v y \end{bmatrix}. \tag{9}$$

An important advantage of the SWF is that its computational cost depends on the number of features in the current window rather than the number of features in the entire map. By varying the spatial or temporal extent of the sliding window, the computational cost of the algorithm can be tailored to fit a given compute envelope, which makes the algorithm suitable for online operation over paths of arbitrary length.

However, the hard cut-off of the SWF may result in only some measurements of a particular feature contributing to the optimization. As a result, the filter may not maximally constrain some vehicle poses, and hence localization may be less accurate than we could expect from the full batch solution.

II. MULTI-STATE CONSTRAINT KALMAN FILTER

The Multi-State Constraint Kalman Filter (MSCKF) [**?**] can be thought of as a hybrid of EKF-SLAM and the SWF. The key idea of the MSCKF is to maintain a sliding window of vehicle poses and to simultaneously update each pose in the window using batch-optimized estimates of features that are visible across the entire window. This update step typically occurs when a tracked feature goes out of view of the camera, but it may also be triggered if the number of vehicle states in the window exceeds some preset threshold.

A. MSCKF State Parametrization

We evaluated the MSCKF using a dataset in which the IMU 'measures' gravity-corrected linear velocities rather than raw linear accelerations (see Section III). In order to accommodate this alternative sensor configuration, the mathematical framework described in this section differs slightly from that described in [**?**].

In our implementation, we parametrize the IMU state at time k as the 13-dimensional vector

$$\mathbf{x}_{I,k} := \begin{bmatrix} \mathbf{q}_{IG,k}^T & \mathbf{b}_{\boldsymbol{\omega},k}^T & \mathbf{b}_{\mathbf{v},k}^T & \mathbf{p}_{G,k}^{IG\ T} \end{bmatrix}^T \tag{10}$$

where $\mathbf{q}_{IG,k}$ is the unit quaternion representing the rotation from the global frame $\mathcal{F}_G$ to the IMU frame $\mathcal{F}_I$, $\mathbf{b}_{\boldsymbol{\omega},k}$ is the bias on the gyro measurements $\boldsymbol{\omega}_m$, $\mathbf{b}_{\mathbf{v},k}$ is the bias on the velocity measurements $\mathbf{v}_m$, and $\mathbf{p}_{G,k}^{IG}$ is the vector from the origin of $\mathcal{F}_G$ to the origin of $\mathcal{F}_I$ expressed in $\mathcal{F}_G$ (i.e., the position of the IMU in the global frame).

At time k, the full state of the MSCKF consists of the current IMU state estimate, and estimates of N 7-dimensional past camera poses in which active feature tracks were visible:

$$\hat{\mathbf{x}}_k := \begin{bmatrix} \hat{\mathbf{x}}_{I,k}^T & \hat{\mathbf{q}}_{C_1 G}^T & \hat{\mathbf{p}}_G^{C_1 G\ T} & \dots & \hat{\mathbf{q}}_{C_N G}^T & \hat{\mathbf{p}}_G^{C_N G\ T} \end{bmatrix}^T \tag{11}$$

We can also define the MSCKF "error state" at time k:

$$\widetilde{\mathbf{x}}_k := \begin{bmatrix} \widetilde{\mathbf{x}}_{I,k}^T & \boldsymbol{\delta\theta}_{C_1}^T & \widetilde{\mathbf{p}}_G^{C_1 G\ T} & \dots & \boldsymbol{\delta\theta}_{C_N}^T & \widetilde{\mathbf{p}}_G^{C_N G\ T} \end{bmatrix}^T \tag{12}$$

where

$$\widetilde{\mathbf{x}}_{I,k} := \begin{bmatrix} \boldsymbol{\delta\theta}_I^T & \widetilde{\mathbf{b}}_{\boldsymbol{\omega},k}^T & \widetilde{\mathbf{b}}_{\mathbf{v},k}^T & \widetilde{\mathbf{p}}_{G,k}^{IG\ T} \end{bmatrix}^T \tag{13}$$

is the 12-dimensional IMU error state. In the above, $\widetilde{x}$ denotes the difference between the true value and the estimated value of the quantity x. The rotational errors $\boldsymbol{\delta\theta}$ are defined according to

$$\boldsymbol{\delta}\mathbf{q} := \hat{\mathbf{q}}^{-1} \otimes \mathbf{q} \simeq \begin{bmatrix} \frac{1}{2}\boldsymbol{\delta\theta}^T & 1 \end{bmatrix}^T. \tag{14}$$

Accordingly, the MSCKF state covariance $\hat{\mathbf{P}}_k$ is a $(12+6N) \times (12+6N)$ matrix that may be partitioned as

$$\hat{\mathbf{P}}_k = \begin{bmatrix} \hat{\mathbf{P}}_{II,k} & \hat{\mathbf{P}}_{IC,k} \\ \hat{\mathbf{P}}_{IC,k}^T & \hat{\mathbf{P}}_{CC,k} \end{bmatrix} \tag{15}$$

where $\hat{\mathbf{P}}_{II,k}$ is the 12×12 covariance matrix of the current IMU state, $\hat{\mathbf{P}}_{CC,k}$ is the $6N \times 6N$ covariance matrix of the camera poses, and $\hat{\mathbf{P}}_{IC,k}$ is the $12 \times 6N$ cross-correlation between the current IMU state and the past camera poses.

B. MSCKF State Augmentation

When a new camera image becomes available, the MSCKF state must be augmented with the current camera pose. We obtain the camera pose by applying the known transformation $\left(\mathbf{q}_{CI}, \mathbf{p}_I^{CI}\right)$ to a copy of the current IMU pose:

$$\hat{\mathbf{q}}_{C_{N+1}G} = \mathbf{q}_{CI} \otimes \hat{\mathbf{q}}_{IG,k} \tag{16}$$

$$\hat{\mathbf{p}}_G^{C_{N+1}G} = \hat{\mathbf{p}}_G^{IG} + \hat{\mathbf{C}}_{IG,k}^T \hat{\mathbf{p}}_I^{CI} \tag{17}$$

where $\hat{\mathbf{C}}_{IG,k}$ is the rotation matrix corresponding to $\hat{\mathbf{q}}_{IG,k}$ and $\otimes$ denotes quaternion multiplication.

Assuming the MSCKF state has already been augmented by N camera poses, we add the $(N+1)^{\text{th}}$ camera pose to the state as follows:

$$\hat{\mathbf{x}}_k \leftarrow \begin{bmatrix} \hat{\mathbf{x}}_k^T & \hat{\mathbf{q}}_{C_{N+1}G}^T & \hat{\mathbf{p}}_G^{C_{N+1}G\ T} \end{bmatrix}^T. \tag{18}$$

We must also augment the MSCKF state covariance:

$$\hat{\mathbf{P}}_k \leftarrow \begin{bmatrix} \mathbf{1}_{12+6N} \\ \mathbf{J}_k \end{bmatrix} \hat{\mathbf{P}}_k \begin{bmatrix} \mathbf{1}_{12+6N} \\ \mathbf{J}_k \end{bmatrix}^T \tag{19}$$

where the Jacobian $\mathbf{J}_k$ is given by

$$\mathbf{J}_k = \begin{bmatrix} \hat{\mathbf{C}}_{CI,k} & \mathbf{0}_{3\times 6} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 6N} \\ \left(\hat{\mathbf{C}}_{IG,k}^T \mathbf{p}_{I,k}^{CI}\right)^{\times} & \mathbf{0}_{3\times 6} & \mathbf{1}_3 & \mathbf{0}_{3\times 6N} \end{bmatrix}. \tag{20}$$

C. IMU State Estimate Propagation

The evolution of the mean estimated IMU state $\hat{\mathbf{x}}_I$ over time is described by a continuous-time motion model:

$$\dot{\hat{\mathbf{q}}}_{IG} = \frac{1}{2}\mathbf{\Omega}\left(\hat{\boldsymbol{\omega}}\right)\hat{\mathbf{q}}_{IG} \tag{21}$$

$$\dot{\hat{\mathbf{b}}}_{\boldsymbol{\omega}} = \mathbf{0}_{3\times 1} \tag{22}$$

$$\dot{\hat{\mathbf{b}}}_{\mathbf{v}} = \mathbf{0}_{3\times 1} \tag{23}$$

$$\dot{\hat{\mathbf{p}}}_G^{IG} = \hat{\mathbf{C}}_{IG}^T \hat{\mathbf{v}} \tag{24}$$

where $\hat{\mathbf{C}}_{IG}$ is the rotation matrix corresponding to $\hat{\mathbf{q}}_{IG}$,

$$\hat{\mathbf{v}} = \mathbf{v}_m - \hat{\mathbf{b}}_{\mathbf{v}}, \qquad \hat{\boldsymbol{\omega}} = \boldsymbol{\omega}_m - \hat{\mathbf{b}}_{\boldsymbol{\omega}},$$

$$\mathbf{\Omega}\left(\hat{\boldsymbol{\omega}}\right) = \begin{bmatrix} -\hat{\boldsymbol{\omega}}^{\times} & \hat{\boldsymbol{\omega}} \\ -\hat{\boldsymbol{\omega}}^T & 0 \end{bmatrix}, \quad \text{and} \quad \hat{\boldsymbol{\omega}}^{\times} = \begin{bmatrix} 0 & -\hat{\omega}_3 & \hat{\omega}_2 \\ \hat{\omega}_3 & 0 & -\hat{\omega}_1 \\ -\hat{\omega}_2 & \hat{\omega}_1 & 0 \end{bmatrix}.$$

In our implementation we propagate the motion model using a simple forward-Euler integration rather than the fifth-order Runge-Kutta procedure used in [**?**].

We can also examine the linearized continuous-time model of the IMU error state:

$$\dot{\tilde{\mathbf{x}}}_I = \mathbf{F}\tilde{\mathbf{x}}_I + \mathbf{G}\mathbf{n}_I \tag{25}$$

where the Jacobians $\mathbf{F}$, $\mathbf{G}$ are given by

$$\mathbf{F} = \begin{bmatrix} -\hat{\boldsymbol{\omega}}^{\times} & -\mathbf{1}_3 & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ -\hat{\mathbf{C}}_{IG}^T \hat{\mathbf{v}}^{\times} & \mathbf{0}_{3\times 3} & -\hat{\mathbf{C}}_{IG}^T & \mathbf{0}_{3\times 3} \end{bmatrix} \tag{26}$$

$$\mathbf{G} = \begin{bmatrix} -\mathbf{1}_3 & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{1}_3 & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{1}_3 \\ \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & -\hat{\mathbf{C}}_{IG}^T & \mathbf{0}_{3\times 3} \end{bmatrix}, \tag{27}$$

and $\mathbf{n}_I = \begin{bmatrix} \mathbf{n}_{\boldsymbol{\omega}}^T & \mathbf{n}_{\mathbf{b}_{\boldsymbol{\omega}}}^T & \mathbf{n}_{\mathbf{v}}^T & \mathbf{n}_{\mathbf{b}_{\mathbf{v}}}^T \end{bmatrix}^T$ is the IMU process noise, which has covariance matrix $\mathbf{Q}_I$.

D. MSCKF State Covariance Propagation

With reference to the partitions defined in (15), we compute the predicted camera-camera and IMU-camera state covariances as follows:

$$\hat{\mathbf{P}}_{CC,k+1}^{-} = \hat{\mathbf{P}}_{CC,k} \tag{28}$$

$$\hat{\mathbf{P}}_{IC,k+1}^{-} = \mathbf{\Phi}\left(t_k + T, t_k\right)\hat{\mathbf{P}}_{IC,k} \tag{29}$$

where T is the IMU sampling period, and the state transition matrix $\mathbf{\Phi}\left(t_k + T, t_k\right)$ is obtained by integrating

$$\dot{\mathbf{\Phi}}\left(t_k + \tau, t_k\right) = \mathbf{F}\mathbf{\Phi}\left(t_k + \tau, t_k\right), \tau \in [0, T] \tag{30}$$

with the initial condition $\mathbf{\Phi}\left(t_k, t_k\right) = \mathbf{1}_{12}$.

We obtain the predicted IMU-IMU state covariance $\hat{\mathbf{P}}_{II,k+1}^{-}$ by integrating

$$\begin{aligned} \dot{\hat{\mathbf{P}}}_{II}\left(t_k + \tau\right) = \mathbf{F}\hat{\mathbf{P}}_{II}\left(t_k + \tau\right)\hat{x}_{C_i}^{C_1 C_i} + \hat{\mathbf{P}}_{II}\left(t_k + \tau\right)\mathbf{F}^T \\ + \mathbf{G}\mathbf{Q}_I\mathbf{G}^T, \tau \in [0, T] \end{aligned} \tag{31}$$

with the initial condition $\hat{\mathbf{P}}_{II}\left(t_k\right) = \hat{\mathbf{P}}_{II,k}$.

E. Feature Position Estimation

When a feature f_j goes out of view of the camera, the MSCKF estimates its position $\hat{\mathbf{p}}_G^{f_j G}$ using an inverse-depth least-squares Gauss-Newton optimization. The procedure takes as input N camera poses and N sets of "ideal" pixel measurements, where "ideal" means that the pixel measurements have been corrected for the camera intrinsics:

$$\hat{\mathbf{z}}_i^{(j)} = \begin{bmatrix} u_i' & v_i' \end{bmatrix}^T = \begin{bmatrix} (u_i - c_u)/f_u & (v_i - c_v)/f_v \end{bmatrix}^T \tag{32}$$

We initialize the optimization by estimating the position of feature f_j in camera frame C_1 using a linear least squares method with measurements from the first two camera frames, C_1 and C_2:

$$\hat{\mathbf{p}}_{C_1}^{f_j C_1} := \begin{bmatrix} \hat{X}_{C_1}^{(j)} & \hat{Y}_{C_1}^{(j)} & \hat{Z}_{C_1}^{(j)} \end{bmatrix}^T = \lambda \boldsymbol{\rho}_{C_1}^{f_j} \tag{33}$$

where

$$\boldsymbol{\rho}_{C_i}^{f_j} := \frac{1}{\sqrt{u_i'^2 + v_i'^2 + 1}} \begin{bmatrix} u_i' & v_i' & 1 \end{bmatrix}^T \tag{34}$$

and

$$\lambda = \left[(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T \hat{\mathbf{p}}_{C_1}^{C_2 C_1}\right]_1 \tag{35}$$

with

$$\mathbf{A} := \begin{bmatrix} \boldsymbol{\rho}_{C_1}^{f_j} & -\boldsymbol{\rho}_{C_2}^{f_j} \end{bmatrix}. \tag{36}$$

Next, we define a vector of three parameters:

$$\hat{\mathbf{y}} = \begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix}^T = \frac{1}{\hat{Z}_{C_1}^{(j)}} \begin{bmatrix} \hat{X}_{C_1}^{(j)} & \hat{Y}_{C_1}^{(j)} & 1 \end{bmatrix}^T \tag{37}$$

The feature position in camera frame C_i can be expressed as

$$\hat{\mathbf{p}}_{C_i}^{f_j C_i} = \hat{\mathbf{C}}_{i1}\hat{\mathbf{p}}_{C_1}^{f_j C_1} + \hat{\mathbf{p}}_{C_i}^{C_1 C_i} \tag{38}$$

With our state parameters, we can use this equation to define three functions:

$$\begin{aligned} \begin{bmatrix} h_1(\hat{\mathbf{y}}) \\ h_2(\hat{\mathbf{y}}) \\ h_3(\hat{\mathbf{y}}) \end{bmatrix} &= \hat{\mathbf{C}}_{i1} \begin{bmatrix} \alpha \\ \beta \\ 1 \end{bmatrix} + \gamma \hat{\mathbf{p}}_{C_i}^{C_1 C_i} \\ &= \begin{bmatrix} C_{11}\alpha + C_{12}\beta + C_{13} + \gamma \hat{x}_{C_i}^{C_1 C_i} \\ C_{21}\alpha + C_{22}\beta + C_{23} + \gamma \hat{y}_{C_i}^{C_1 C_i} \\ C_{31}\alpha + C_{32}\beta + C_{33} + \gamma \hat{z}_{C_i}^{C_1 C_i} \end{bmatrix} \end{aligned} \tag{39}$$

The camera measurement error can then be written as

$$\mathbf{e}(\hat{\mathbf{y}}) = \hat{\mathbf{z}}_i^{(j)} - \frac{1}{h_3(\hat{\mathbf{y}})} \begin{bmatrix} h_1(\hat{\mathbf{y}}) \\ h_2(\hat{\mathbf{y}}) \end{bmatrix} \tag{40}$$

The least squares linear system then becomes

$$(\mathbf{E}^T \mathbf{W}^{-1} \mathbf{E}) \delta \mathbf{y}^* = -\mathbf{E}^T \mathbf{W}^{-1} \mathbf{e}(\hat{\mathbf{y}}) \tag{41}$$

where

$$\mathbf{E} = \frac{\partial \mathbf{e}}{\partial \hat{\mathbf{y}}} \quad \text{and} \quad \mathbf{W} = \begin{bmatrix} \mathbf{R}_1'^{(j)} & & \\ & \ddots & \\ & & \mathbf{R}_N'^{(j)} \end{bmatrix} \tag{42}$$

with $\mathbf{R}_i'^{(j)} = \text{diag}\left\{\sigma_{u'}^2, \sigma_{v'}^2\right\}$.

The Jacobian elements are given by

$$\frac{\partial \mathbf{e}}{\partial \alpha} = \frac{1}{h_3^2} \begin{bmatrix} -\frac{\partial h_1}{\partial \alpha} h_3 + \frac{\partial h_3}{\partial \alpha} h_1 \\ -\frac{\partial h_2}{\partial \alpha} h_3 + \frac{\partial h_3}{\partial \alpha} h_2 \end{bmatrix} \tag{43}$$

$$\frac{\partial \mathbf{e}}{\partial \beta} = \frac{1}{h_3^2} \begin{bmatrix} -\frac{\partial h_1}{\partial \beta} h_3 + \frac{\partial h_3}{\partial \beta} h_1 \\ -\frac{\partial h_2}{\partial \beta} h_3 + \frac{\partial h_3}{\partial \beta} h_2 \end{bmatrix} \tag{44}$$

$$\frac{\partial \mathbf{e}}{\partial \gamma} = \frac{1}{h_3^2} \begin{bmatrix} -\frac{\partial h_1}{\partial \gamma} h_3 + \frac{\partial h_3}{\partial \gamma} h_1 \\ -\frac{\partial h_2}{\partial \gamma} h_3 + \frac{\partial h_3}{\partial \gamma} h_2 \end{bmatrix} \tag{45}$$

with

$$\begin{aligned} \frac{\partial h_1}{\partial \alpha} &= C_{11}, & \frac{\partial h_1}{\partial \beta} &= C_{12}, & \frac{\partial h_1}{\partial \gamma} &= \hat{x}_{C_i}^{C_1 C_i} \\ \frac{\partial h_2}{\partial \alpha} &= C_{21}, & \frac{\partial h_2}{\partial \beta} &= C_{22}, & \frac{\partial h_2}{\partial \gamma} &= \hat{y}_{C_i}^{C_1 C_i} \\ \frac{\partial h_3}{\partial \alpha} &= C_{31}, & \frac{\partial h_3}{\partial \beta} &= C_{32}, & \frac{\partial h_3}{\partial \gamma} &= \hat{z}_{C_i}^{C_1 C_i}. \end{aligned} \tag{46}$$

F. MSCKF State Correction Equations

Now that we have estimated the positions of any features that have gone out of view, we can apply the corresponding motion constraints to the window of poses from which each feature was observed. We begin by forming the exteroceptive measurement error corresponding to an observation $\mathbf{z}_i^{(j)}$ of feature f_j from the i^{th} camera pose C_i in the window:

$$\mathbf{r}_i^{(j)} := \mathbf{z}_i^{(j)} - \hat{\mathbf{z}}_i^{(j)} \tag{47}$$

where

$$\hat{\mathbf{z}}_i^{(j)} = \frac{1}{\hat{Z}_{C_i}^{(j)}} \begin{bmatrix} \hat{X}_{C_i}^{(j)} & \hat{Y}_{C_i}^{(j)} \end{bmatrix}^T \tag{48}$$

with

$$\hat{\mathbf{p}}_{C_i}^{f_j C_i} = \begin{bmatrix} \hat{X}_{C_i}^{(j)} & \hat{Y}_{C_i}^{(j)} & \hat{Z}_{C_i}^{(j)} \end{bmatrix}^T = \hat{\mathbf{C}}_{C_i G} \left(\hat{\mathbf{p}}_G^{f_j G} - \hat{\mathbf{p}}_G^{C_i G} \right). \tag{49}$$

If we linearize (47) about the estimates for the camera pose and feature position, we obtain an estimate of the exteroceptive measurement error

$$\mathbf{r}_i^{(j)} \simeq \mathbf{H}_{\mathbf{x},i}^{(j)} \widetilde{\mathbf{x}}_i + \mathbf{H}_{f,i}^{(j)} \widetilde{\mathbf{p}}_G^{f_j G} + \mathbf{n}_i^{(j)} \tag{50}$$

where $\mathbf{H}_{\mathbf{x},i}^{(j)}$ and $\mathbf{H}_{f,i}^{(j)}$ are the Jacobians of the measurement of feature f_j from camera pose i with respect to the filter state and the position of the feature, respectively. These are given by

$$\mathbf{H}_{\mathbf{x},i}^{(j)} = \begin{bmatrix} \mathbf{0} & \mathbf{J}_i^{(j)} \left(\hat{\mathbf{p}}_{C_i}^{f_j C_i} \right)^{\times} & -\mathbf{J}_i^{(j)} \hat{\mathbf{C}}_{C_i G} & \mathbf{0} \end{bmatrix} \tag{51}$$

$$\mathbf{H}_{f,i}^{(j)} = \mathbf{J}_i^{(j)} \hat{\mathbf{C}}_{C_i G} \tag{52}$$

where the left $\mathbf{0}$ in $\mathbf{H}_{\mathbf{x},i}^{(j)}$ has dimension $2 \times (12 + 6\,(i-1))$, the right $\mathbf{0}$ has dimension $2 \times 6\,(N-i)$, and

$$\mathbf{J}_i^{(j)} = \frac{1}{\left(\hat{Z}_{C_i}^{(j)}\right)^2} \begin{bmatrix} \hat{Z}_{C_i}^{(j)} & 0 & -\hat{X}_{C_i}^{(j)} \\ 0 & \hat{Z}_{C_i}^{(j)} & -\hat{Y}_{C_i}^{(j)} \end{bmatrix}. \tag{53}$$

$\mathbf{n}_i^{(j)}$ is a zero-mean Gaussian noise term with covariance matrix $\mathbf{R}_i^{(j)} = \text{diag}\left\{\sigma_u^2, \sigma_v^2\right\}$.

By stacking the errors $\mathbf{r}_i^{(j)}$, we arrive at an expression for the $2M_j \times 1$ measurement error vector of feature f_j over the entire window of camera poses, where M_j is the number of camera poses from which feature f_j was observed:

$$\mathbf{r}^{(j)} \simeq \mathbf{H}_{\mathbf{x}}^{(j)} \widetilde{\mathbf{x}} + \mathbf{H}_f^{(j)} \widetilde{\mathbf{p}}_G^{f_j G} + \mathbf{n}^{(j)} \tag{54}$$

The noise vector $\mathbf{n}^{(j)}$ has covariance matrix $\mathbf{R}^{(j)} = \text{diag}\left\{\mathbf{R}_1^{(j)}, \ldots, \mathbf{R}_{M_j}^{(j)}\right\}$.

However, the EKF assumes that measurement errors are linear in the error in the state and have an additive zero-mean Gaussian noise component that is *uncorrelated* to the error in the state. Since the error in the feature positions *is* correlated to the error in the state, $\mathbf{r}^{(j)}$ cannot be used in the EKF directly. Figure 1 shows the effect of attempting to use this error directly in the EKF. The correlation between the measurement error on the state causes the filter to behave very poorly after a small number of time steps.

In order to transform $\mathbf{r}^{(j)}$ into a usable form for the EKF, we can define a semi-unitary matrix $\mathbf{A}$ whose columns form the basis of the left nullspace of $\mathbf{H}_f^{(j)}$, and project $\mathbf{r}^{(j)}$ into this nullspace to obtain an error equation of the correct form:

$$\begin{aligned} \mathbf{r}_o^{(j)} &:= \mathbf{A}^T \left(\mathbf{z}_i^{(j)} - \hat{\mathbf{z}}_i^{(j)} \right) && (55) \\ &\simeq \mathbf{A}^T \mathbf{H}_{\mathbf{x}}^{(j)} \widetilde{\mathbf{x}} + \mathbf{0} + \mathbf{A}^T \mathbf{n}^{(j)} && (56) \\ &=: \mathbf{H}_o^{(j)} \widetilde{\mathbf{x}} + \mathbf{n}_o^{(j)} && (57) \end{aligned}$$

Since $\mathbf{H}_f^{(j)}$ has full column rank, $\mathbf{A}$ has dimension $2M_j \times (2M_j - 3)$ and $\mathbf{r}_o^{(j)}$ has dimension $(2M_j - 3) \times 1$. The covariance matrix of $\mathbf{n}_o^{(j)}$ is given by $\mathbf{R}_o^{(j)} = \mathbf{A}^T \mathbf{R}^{(j)} \mathbf{A}$.

We can now stack all the errors $\mathbf{r}_o^{(j)}$ for all the features in the current batch to arrive at

$$\mathbf{r}_o = \mathbf{H}_o \widetilde{\mathbf{x}} + \mathbf{n}_o. \tag{58}$$

The dimension of this vector can be quite large in practice, so we use the QR-decomposition of $\mathbf{H}_o$ to reduce the computational complexity of the EKF update:

$$\mathbf{H}_o = \begin{bmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 \end{bmatrix} \begin{bmatrix} \mathbf{T_H} \\ \mathbf{0} \end{bmatrix} \tag{59}$$

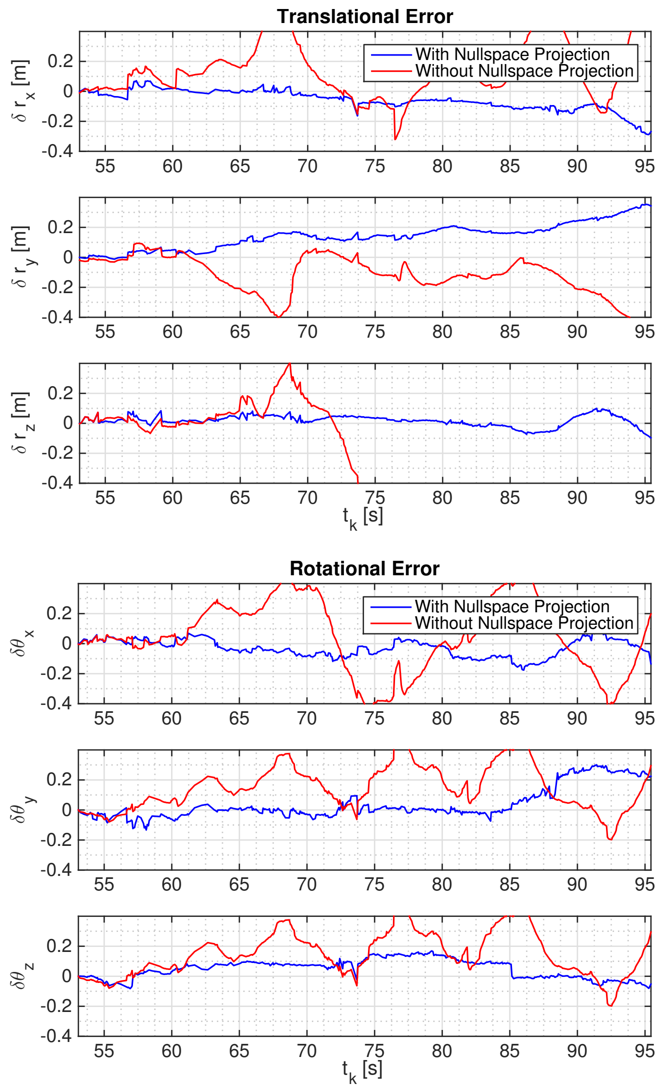

Fig. 1. Effect of nullspace projection on MSCKF performance

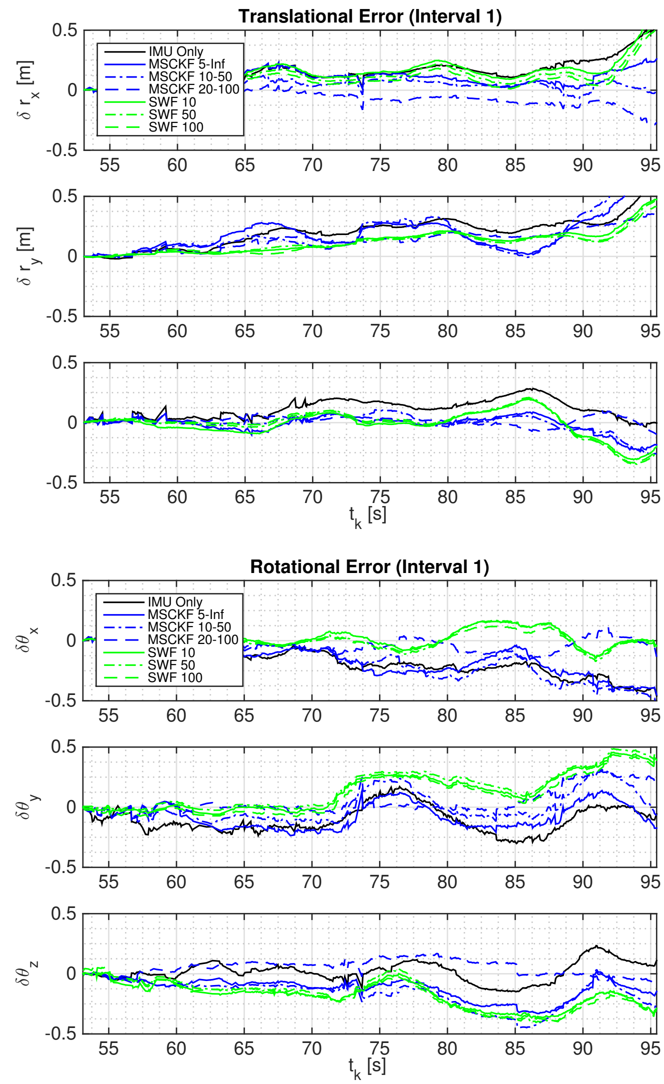

Fig. 2. MSCKF vs. SWF

where $\mathbf{Q}_1, \mathbf{Q}_2$ are unitary matrices and $\mathbf{T_H}$ is an upper-triangular matrix. Substituting this result into (58) and pre-multiplying by $\begin{bmatrix}\mathbf{Q}_1 & \mathbf{Q}_2\end{bmatrix}^T$, we obtain

$$\begin{bmatrix}\mathbf{Q}_1^T\mathbf{r}_o \\ \mathbf{Q}_2^T\mathbf{r}_o\end{bmatrix} = \begin{bmatrix}\mathbf{T_H} \\ \mathbf{0}\end{bmatrix}\widetilde{\mathbf{x}} + \begin{bmatrix}\mathbf{Q}_1^T\mathbf{n}_o \\ \mathbf{Q}_2^T\mathbf{n}_o\end{bmatrix}. \tag{60}$$

Noting that the quantity $\mathbf{Q}_2^T\mathbf{r}_o$ is only noise, we discard it and define a new error term that we use in the EKF update:

$$\mathbf{r}_n := \mathbf{Q}_1^T\mathbf{r}_o = \mathbf{T_H}\widetilde{\mathbf{x}} + \mathbf{Q}_1^T\mathbf{n}_o =: \mathbf{T_H}\widetilde{\mathbf{x}} + \mathbf{n}_n \tag{61}$$

The covariance matrix of $\mathbf{n}_n$ is given by $\mathbf{R}_n = \mathbf{Q}_1^T\mathbf{R}_o^{(j)}\mathbf{Q}_1$.

Finally, we can formulate the Kalman gain and correction equations to arrive at the updated estimates for the filter state and covariance:

$$\mathbf{K} = \hat{\mathbf{P}}_{k+1}^{-}\mathbf{T}_H^T\left(\mathbf{T}_H\hat{\mathbf{P}}_{k+1}^{-}\mathbf{T}_H^T + \mathbf{R}_n\right)^{-1} \tag{62}$$

$$\Delta\mathbf{x}_{k+1} = \mathbf{K}\mathbf{r}_n \tag{63}$$

$$\begin{aligned}\hat{\mathbf{P}}_{k+1} = {} & \left(\mathbf{1}_{12+6N} - \mathbf{K}\mathbf{T}_H\right)\hat{\mathbf{P}}_{k+1}^{-}\left(\mathbf{1}_{12+6N} - \mathbf{K}\mathbf{T}_H\right)^T \\ & + \mathbf{K}\mathbf{R}_n\mathbf{K}^T.\end{aligned} \tag{64}$$

III. EXPERIMENTS

We conducted a comparative study of the MSCKF and SWF algorithms using the “Starry Night” dataset from the University of Toronto Institute for Aerospace Studies (UTIAS). The dataset consists of a rigidly attached stereo camera and IMU (Figure 4) observing a set of 20 features while moving along an arbitrary 3D path. The dataset is well-suited to evaluating SLAM algorithms since accurate ground truth from a Vicon motion capture system is available for both the sensor head motion and the feature positions (Figure 5). Since our algorithms are designed to make use of a monocular camera, we artificially blinded the stereo camera by using measurements from the left camera only.

Things quickly got out of hand when Matlab achieved sentience. We were forced to abandon our lab and have been surviving in the harsh environment of the MarsDome for weeks. If this ever gets published, please send help!

IV. CONCLUSIONS

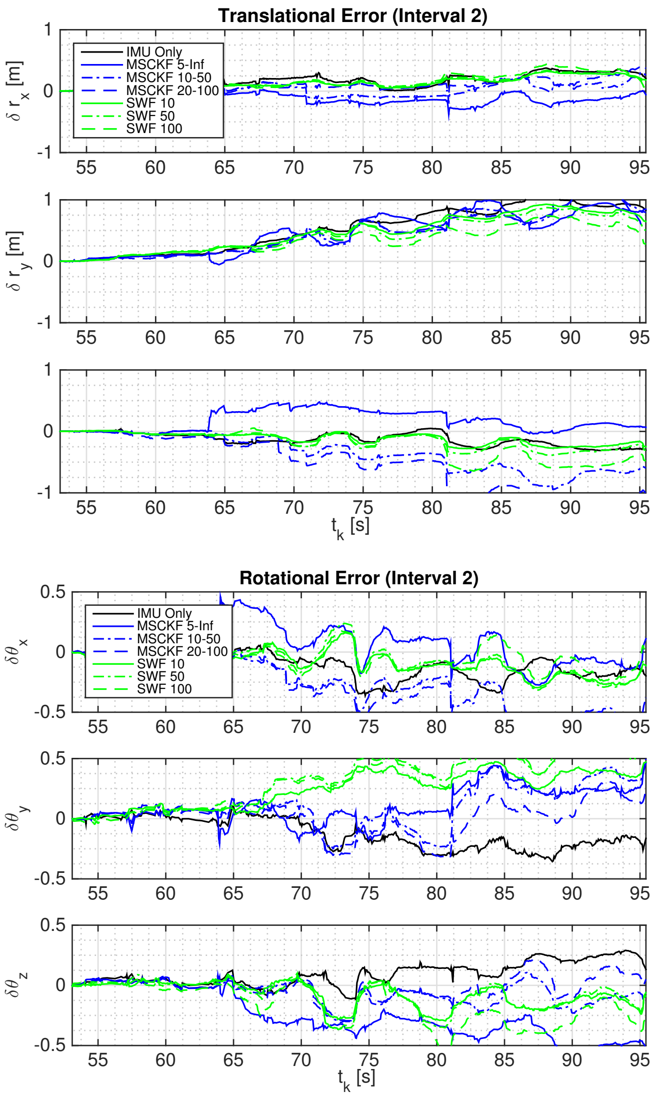

Fig. 3. MSCKF vs. SWF

Fig. 4. The sensor head used in our experiments. The IMU measures translational and rotational velocities, while the stereo camera measures the positions of point features. In our experiments, we artificially blinded the stereo camera by using measurements from the left camera only.

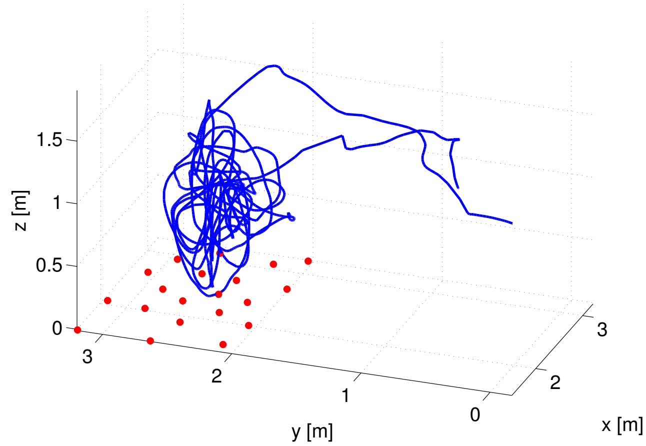

Fig. 5. Vicon ground truth for sensor head motion (blue) and feature positions (red) in the "Starry Night" dataset.