# The Battle for Filter Supremacy: A Comparative Study of the Multi-State Constraint Kalman Filter and the Sliding Window Filter

Lee Clement[1] and Valentin Peretroukhin[1]

***Abstract*— Accurate and consistent egomotion estimation is an important part of autonomous navigation. For this task, the combination of visual and inertial sensors is an inexpensive, compact, and complementary hardware suite that can be used on many ground and aerial vehicles. In this work, we compare two modern approaches to egomotion estimation: the Multi-State Constraint Kalman Filter (MSCKF) and the Sliding Window Filter (SWF). Both filters use an Inertial Measurement Unit (IMU) to estimate the motion of a vehicle and then correct this motion with observations of salient features from a monocular camera. While the SWF estimates feature positions as part of the filter state itself, the MSCKF optimizes feature positions in a separate procedure without including them in the filter state. We present experimental comparisons of the performance of the MSCKF and SWF on several datasets with real and synthetic features. In particular, we compare the accuracy and consistency of the two filters, and analyze the effect of feature track length and feature density on the performance of each filter. In general, our results show the SWF to be more accurate than the MSCKF. However, the MSCKF is significantly more computationally efficient, has good consistency properties, and improves in accuracy as more features are tracked.**

## 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 because 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, IMU data 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.

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


Fig. 1. 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.

In this work, we compare and contrast two modern solutions the Sliding Window Filter (SWF), the Multi-State Constraint Kalman Filter (MSCKF) [1], [2]. The MSCKF is a hybrid approach that combines the computational efficiency of an Extended Kalman Filter (EKF) with the accuracy of the SWF. The MSCKF has recently been used successfully in real-time, accurate position tracking onboard a smartphone [3], while the SWF has been applied to planetary landing [4].

We present an experimental comparison of the SWF and MSCKF using different datasets consisting of IMU and camera data with accurate ground truth for both sensor motion and landmark positions. To motivate the MSCKF, we first outline a more commmon approach of the EKF.

## II. 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.

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.

## III. 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 (GN) optimization which acts on a sliding window of $K$ poses and $N$ features:

$$\mathbf{x} := \begin{bmatrix} \mathbf{x}_0^T & \dots & \mathbf{x}_K^T & \mathbf{f}_1^T & \dots & \mathbf{f}_N^T \end{bmatrix}^T. \tag{1}$$

The optimization applies an update, $\delta\mathbf{x}^*$, by solving the linear system:

$$(\mathbf{H}^T\mathbf{T}^{-1}\mathbf{H})\delta\mathbf{x}^* = -\mathbf{H}^T\mathbf{T}^{-1}\mathbf{e}(\bar{\mathbf{x}}) \tag{2}$$

The matrix $\mathbf{H}$ is given by (showing non-zero blocks only)

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

where

$$\mathbf{G}_{\mathbf{x},k} = \begin{bmatrix} \mathbf{G}_{\mathbf{x},k}^1 & \dots & \mathbf{G}_{\mathbf{x},k}^M \end{bmatrix}^T \tag{4}$$

$$\mathbf{G}_{\mathbf{x},k}^j = \frac{\partial \mathbf{g}}{\partial \mathbf{p}}\Big|_{\bar{\mathbf{p}}_{C_k}^{f_j C_k}} \begin{bmatrix} -\mathbf{C}_{CG}\mathbf{C}_{IG,k} \\ \mathbf{C}_{CG}(\mathbf{C}_{IG,k}(\boldsymbol{\rho}_i^{f_j G} - \mathbf{r}_G^{IG,k}))^\times \end{bmatrix}^T \tag{5}$$

$$\mathbf{G}_{\mathbf{f},k} = \underbrace{\begin{bmatrix} \mathbf{G}_{\mathbf{f},k}^1 & & & \\ & \mathbf{G}_{\mathbf{f},k}^2 & & \\ & & \vdots & \\ & & & \mathbf{G}_{\mathbf{f},k}^M \end{bmatrix}}_{j\text{th block column position given by feature ID } j} \tag{6}$$

$$\mathbf{G}_{\mathbf{f},k}^j = \frac{\partial \mathbf{g}}{\partial \mathbf{p}}\Big|_{\bar{\mathbf{p}}_{C_k}^{f_j C_k}} \mathbf{C}_{CI}\mathbf{C}_{IG,k} \tag{7}$$

with

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

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

The weight matrix $\mathbf{T}$ is given by:

$$\mathbf{T} := \begin{bmatrix} \mathbf{T}_1 & & \\ & \ddots & \\ & & \mathbf{T}_k \end{bmatrix} \tag{10}$$

where

$$\mathbf{T}_k := \begin{bmatrix} \mathbf{H}_{\mathbf{w},k}\mathbf{Q}_k\mathbf{H}_{\mathbf{w},k}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_{\mathbf{n},k}\mathbf{R}_k\mathbf{G}_{\mathbf{n},k}^T \end{bmatrix}, \tag{11}$$

and $\mathbf{R}_k$, $\mathbf{Q}_k$ are observation and motion model noise covariances, and $\mathbf{G}_{\mathbf{n},k}$, $\mathbf{H}_{\mathbf{w},k}$ are the observation and motion model Jacobians with respect to the noise.

The update is performed repeatedly, until convergence (in this work, we use the criterion $|\delta\mathbf{x}^*| < 10^{-3}$). 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.

## IV. Multi-State Constraint Kalman Filter

The Multi-State Constraint Kalman Filter (MSCKF) [1], [2] 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 V). In order to accommodate this alternative sensor configuration, the mathematical framework described in this section differs slightly from that described in [2].

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

$$\mathbf{x}_{I,k} := \begin{bmatrix} \mathbf{q}_{IG,k}^T & \mathbf{b}_{\boldsymbol{\omega},k}^T & \mathbf{b}_{\mathbf{v},k}^T & \mathbf{p}_{G,k}^{IG\,T} \end{bmatrix}^T \tag{12}$$

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:

$$\hat{\mathbf{x}}_k := \begin{bmatrix} \hat{\mathbf{x}}_{I,k}^T & \hat{\mathbf{q}}_{C_1G}^T & \hat{\mathbf{p}}_G^{C_1G\,T} & \dots & \hat{\mathbf{q}}_{C_NG}^T & \hat{\mathbf{p}}_G^{C_NG\,T} \end{bmatrix}^T.$$

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

$$\widetilde{\mathbf{x}}_k := \begin{bmatrix} \widetilde{\mathbf{x}}_{I,k}^T & \boldsymbol{\delta\theta}_{C_1}^T & \widetilde{\mathbf{p}}_G^{C_1G\,T} & \dots & \boldsymbol{\delta\theta}_{C_N}^T & \widetilde{\mathbf{p}}_G^{C_NG\,T} \end{bmatrix}^T$$

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

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

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 $12\times 12$ 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:

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

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

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

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

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

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

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

In our implementation we propagate the motion model using a simple forward-Euler integration rather than the fifth-order Runge-Kutta procedure used in [2].

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

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

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:

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

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

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

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

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

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

$$\begin{aligned}\dot{\hat{\mathbf{P}}}_{II}\left(t_k+\tau\right) = \mathbf{F}\hat{\mathbf{P}}_{II}\left(t_k+\tau\right)\hat{x}_{C_i}^{C_1C_i} + \hat{\mathbf{P}}_{II}\left(t_k+\tau\right)\mathbf{F}^T \\ + \mathbf{G}\mathbf{Q}_I\mathbf{G}^T, \tau\in[0,T]\end{aligned} \tag{31}$$

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 $\hat{\mathbf{p}}_G^{f_jG}$ 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:

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

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 $C_2$:

$$\hat{\mathbf{p}}_{C_1}^{f_jC_1} := \begin{bmatrix} \hat{X}_{C_1}^{(j)} & \hat{Y}_{C_1}^{(j)} & \hat{Z}_{C_1}^{(j)} \end{bmatrix}^T = \lambda \boldsymbol{\rho}_{C_1}^{f_j} \quad (33)$$

where

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

and

$$\lambda = \left[ (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T \hat{\mathbf{p}}_{C_1}^{C_2C_1} \right]_1 \quad (35)$$

with

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

Next, we define a vector of three parameters:

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

We can express the feature position in camera frame $C_i$ in terms of its position in camera frame $C_1$:

$$\hat{\mathbf{p}}_{C_i}^{f_jC_i} = \hat{\mathbf{C}}_{i1}\hat{\mathbf{p}}_{C_1}^{f_jC_1} + \hat{\mathbf{p}}_{C_i}^{C_1C_i} \quad (38)$$

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

$$\begin{aligned} \begin{bmatrix} h_1(\hat{\mathbf{y}}) \\ h_2(\hat{\mathbf{y}}) \\ h_3(\hat{\mathbf{y}}) \end{bmatrix} &= \hat{\mathbf{C}}_{i1} \begin{bmatrix} \alpha \\ \beta \\ 1 \end{bmatrix} + \gamma \hat{\mathbf{p}}_{C_i}^{C_1C_i} \\ &= \begin{bmatrix} C_{11}\alpha + C_{12}\beta + C_{13} + \gamma \hat{x}_{C_i}^{C_1C_i} \\ C_{21}\alpha + C_{22}\beta + C_{23} + \gamma \hat{y}_{C_i}^{C_1C_i} \\ C_{31}\alpha + C_{32}\beta + C_{33} + \gamma \hat{z}_{C_i}^{C_1C_i} \end{bmatrix} \end{aligned} \quad (39)$$

The camera measurement error can then be written as

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

The least squares linear system then becomes

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

where

$$\mathbf{E} = \frac{\partial \mathbf{e}}{\partial \hat{\mathbf{y}}} \quad \text{and} \quad \mathbf{W} = \text{diag}\left\{ \mathbf{R}_1^{(j)}, \ldots, \mathbf{R}_N^{(j)} \right\} \quad (42)$$

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

The Jacobian elements are given by

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

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

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

with

$$\begin{aligned} \frac{\partial h_1}{\partial \alpha} &= C_{11}, & \frac{\partial h_1}{\partial \beta} &= C_{12}, & \frac{\partial h_1}{\partial \gamma} &= \hat{x}_{C_i}^{C_1C_i} \\ \frac{\partial h_2}{\partial \alpha} &= C_{21}, & \frac{\partial h_2}{\partial \beta} &= C_{22}, & \frac{\partial h_2}{\partial \gamma} &= \hat{y}_{C_i}^{C_1C_i} \\ \frac{\partial h_3}{\partial \alpha} &= C_{31}, & \frac{\partial h_3}{\partial \beta} &= C_{32}, & \frac{\partial h_3}{\partial \gamma} &= \hat{z}_{C_i}^{C_1C_i}. \end{aligned} \quad (46)$$

### 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 $i^{\text{th}}$ camera pose $C_i$ in the window:

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

where

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

with

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

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

$$\mathbf{r}_i^{(j)} \simeq \mathbf{H}_{\mathbf{x},i}^{(j)} \tilde{\mathbf{x}}_i + \mathbf{H}_{f,i}^{(j)} \tilde{\mathbf{p}}_G^{f_jG} + \mathbf{n}_i^{(j)} \quad (50)$$

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

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

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

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

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

$\mathbf{n}_i^{(j)}$ 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:

$$\mathbf{r}^{(j)} = \mathbf{z}^{(j)} - \hat{\mathbf{z}}^{(j)} \simeq \mathbf{H}_{\mathbf{x}}^{(j)}\tilde{\mathbf{x}} + \mathbf{H}_f^{(j)}\tilde{\mathbf{p}}_G^{f_j G} + \mathbf{n}^{(j)} \tag{54}$$

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 state. Since the term $\mathbf{H}_f^{(j)}\tilde{\mathbf{p}}_G^{f_j G}$ *is* correlated to the state, $\mathbf{r}^{(j)}$ is not of the correct form for the EKF and must be modified to de-correlate it from the state (note that the SWF does not have this problem because feature positions are part of the state itself, and optimized jointly with the poses). Figure 2 shows the effect of attempting to use $\mathbf{r}^{(j)}$ directly in the EKF. The correlation between the measurement noise and the state causes the filter estimates to drift further from the true values than with a modified, de-correlated measurement error.

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 (to first order):

$$\begin{aligned}\mathbf{r}_o^{(j)} &:= \mathbf{A}^T\mathbf{r}^{(j)} \simeq \mathbf{A}^T\mathbf{H}_{\mathbf{x}}^{(j)}\tilde{\mathbf{x}} + \mathbf{0} + \mathbf{A}^T\mathbf{n}^{(j)} \\ &=: \mathbf{H}_o^{(j)}\tilde{\mathbf{x}} + \mathbf{n}_o^{(j)}\end{aligned} \tag{55}$$

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

$$\mathbf{r}_o = \mathbf{H}_o\tilde{\mathbf{x}} + \mathbf{n}_o. \tag{56}$$

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:

$$\mathbf{H}_o = \begin{bmatrix}\mathbf{Q}_1 & \mathbf{Q}_2\end{bmatrix}\begin{bmatrix}\mathbf{T_H} \\ \mathbf{0}\end{bmatrix} \tag{57}$$

where $\mathbf{Q}_1, \mathbf{Q}_2$ are unitary matrices and $\mathbf{T_H}$ is an upper-triangular matrix. Substituting this result into (56) and pre-multiplying by $\begin{bmatrix}\mathbf{Q}_1 & \mathbf{Q}_2\end{bmatrix}^T$, we obtain

$$\begin{bmatrix}\mathbf{Q}_1^T\mathbf{r}_o \\ \mathbf{Q}_2^T\mathbf{r}_o\end{bmatrix} = \begin{bmatrix}\mathbf{T_H} \\ \mathbf{0}\end{bmatrix}\tilde{\mathbf{x}} + \begin{bmatrix}\mathbf{Q}_1^T\mathbf{n}_o \\ \mathbf{Q}_2^T\mathbf{n}_o\end{bmatrix}. \tag{58}$$

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:

$$\mathbf{r}_n := \mathbf{Q}_1^T\mathbf{r}_o = \mathbf{T_H}\tilde{\mathbf{x}} + \mathbf{Q}_1^T\mathbf{n}_o =: \mathbf{T_H}\tilde{\mathbf{x}} + \mathbf{n}_n \tag{59}$$

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

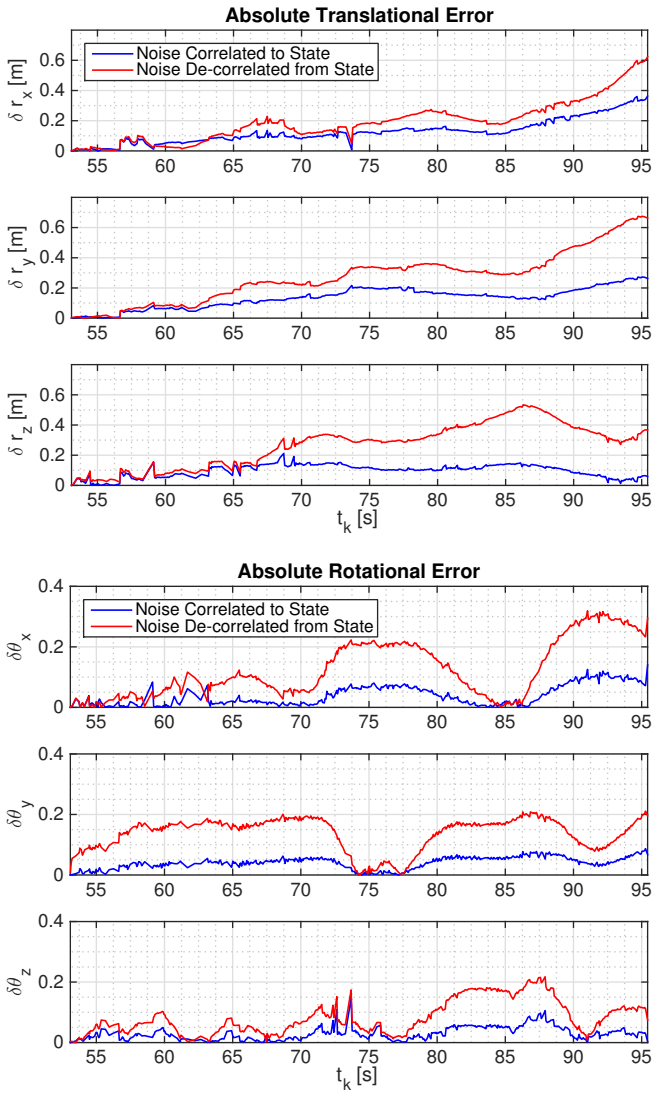


Fig. 2. Effect of correlation between measurement noise and filter state on MSCKF performance. The correlation between the measurement error and the state causes the filter estimates to drift further from the true values than with a modified, de-corellated measurement error.

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

$$\mathbf{K} = \hat{\mathbf{P}}_{k+1}^- \mathbf{T}_H^T \left(\mathbf{T}_H \hat{\mathbf{P}}_{k+1}^- \mathbf{T}_H^T + \mathbf{R}_n\right)^{-1} \tag{60}$$

$$\Delta\mathbf{x}_{k+1} = \mathbf{K}\mathbf{r}_n \tag{61}$$

$$\begin{aligned}\hat{\mathbf{P}}_{k+1} = &\left(\mathbf{1}_{12+6N} - \mathbf{K}\mathbf{T}_H\right)\hat{\mathbf{P}}_{k+1}^- \left(\mathbf{1}_{12+6N} - \mathbf{K}\mathbf{T}_H\right)^T \\ &+ \mathbf{K}\mathbf{R}_n\mathbf{K}^T.\end{aligned} \tag{62}$$

## V. EXPERIMENTS

We compared the performance of the MSCKF and SWF algorithms on two intervals in 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 1) observing a set of 20 features while moving along an arbitrary 3D path. This

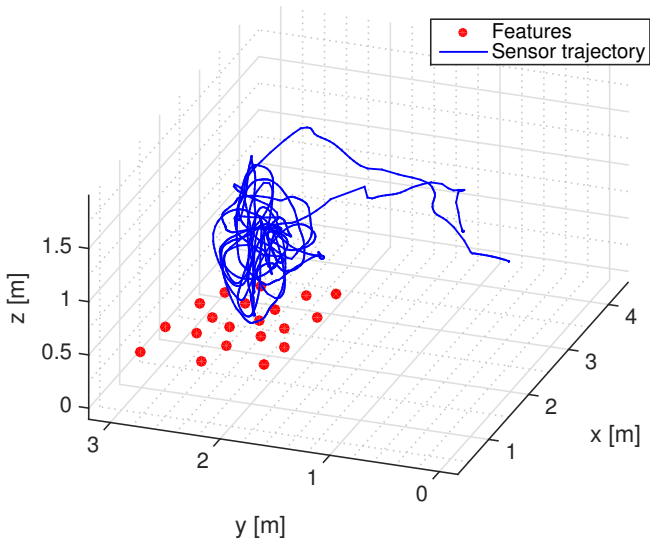


Fig. 3. Vicon ground truth for sensor head motion and feature positions in the "Starry Night" dataset.

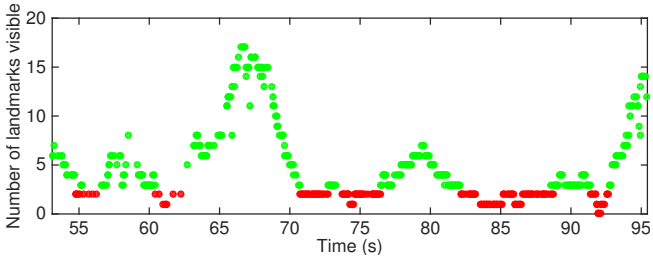


Fig. 4. Number of visible features in the interval $k \in [500, 1000]$. Red points indicate that fewer than three features were visible, while green points indicate at least three features were visible.

dataset is well-suited to evaluating SLAM algorithms since accurate ground truth from a Vicon motion capture system was available for both the sensor head motion and the feature positions (Figure 3). 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.

We implemented and tested both algorithms on a Mac-Book Pro with an Intel i7 processor and 16 GB of RAM running MATLAB 2014b. For both algorithms, we bootstrapped the filter window with one ground truth pose. For the SWF, we initialized feature positions using two-view triangulation.

### A. Experiment 1: Many visible features

In the first experiment, we compared the SWF and MSCKF for various parameter settings on an interval with many visible features. Figure 4 shows the number of visible features in the first interval. At least three features are visible in 64% of this interval, and the number of visible features is frequently in excess of ten.

Figure 5 shows absolute translational and rotational errors for pure IMU integration and various parameter settings for the MSCKF and SWF. For the MSCKF, we varied both the maximum number of observations before triggering an

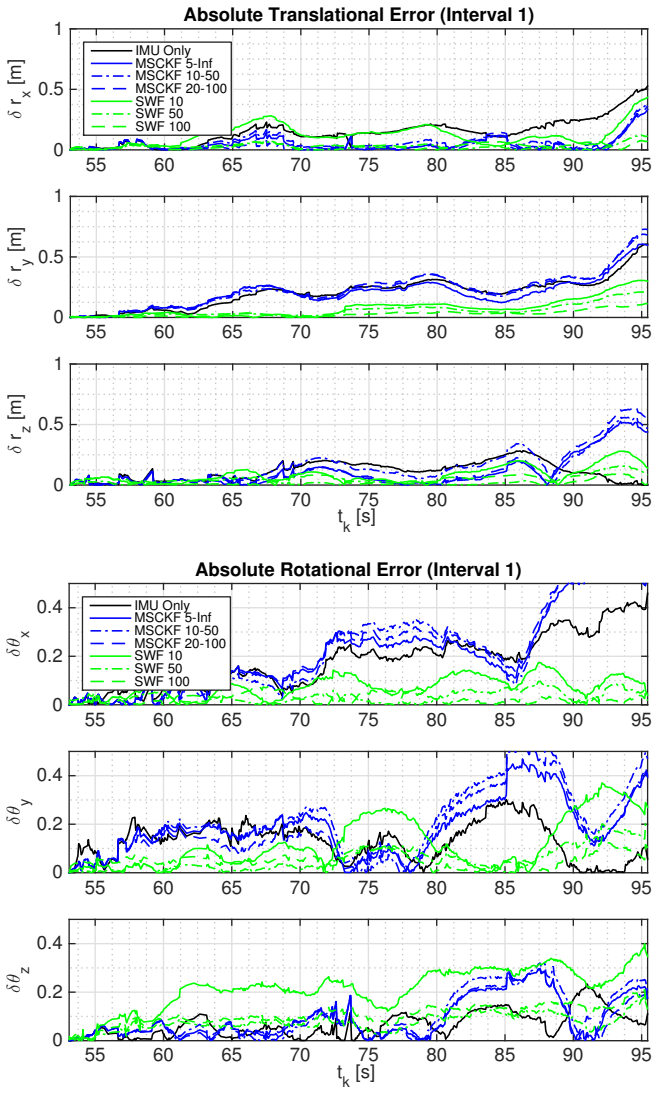


Fig. 5. Comparison of MSCKF, SWF, and IMU integration for multiple parameter settings on the dataset interval $k \in [500, 1000]$. The numbers next to "MSCKF" in the legend refer to the minimum and maximum feature track lengths before an EKF update is triggered, while the number next to "SWF" refers to the number of states in the sliding window.

update and the minimum number of observations for a feature track to be used in an update. For the SWF, we varied the number of states in the sliding window. These parameters did not significantly affect the accuracy of either filter, however Figure 5 shows some small gains in accuracy for larger feature track lengths and window sizes.

On this interval, the SWF slightly outperforms the MSCKF in terms of translational accuracy in the $y$- and $z$-directions, and significantly outperforms the MSCKF in terms of rotational accuracy about the $x$- and $y$-axes. The MSCKF does not perform much better than pure IMU integration on this interval, likely due to the overall low number of feature tracks in this dataset.

### B. Experiment 2: Fewer visible features

In the second experiment, we compared the SWF and MSCKF for various parameter settings on an interval with

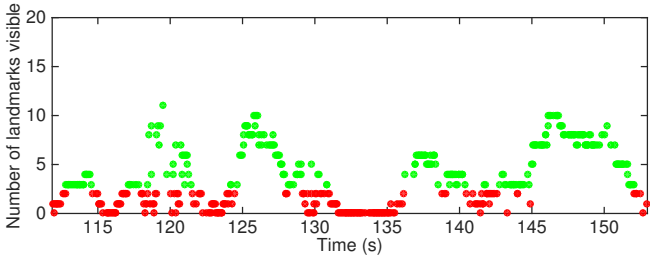


Fig. 6. Number of visible features in the interval $k \in [1215, 1715]$. Red points indicate that fewer than three features were visible, while green points indicate at least three features were visible.

few visible features. Figure 6 shows the number of visible features in the second interval. At least three features are visible in only $56\%$ of this interval, and the number of visible features rarely exceeds ten. Moreover, the features that are visible do not remain in view of the camera for as long a time as in the first interval.

Figure 7 shows absolute translational and rotational errors for pure IMU integration and the same MSCKF and SWF parameter settings as in Experiment 1. Compared to Interval 1, the difference in performance between the MSCKF and SWF is less pronounced, with the exception of the SWF's clearly superior translational accuracy in the $y$-direction. Neither filter performs substantially better than pure IMU integration on this interval, again likely due to the overall low number of feature tracks in this dataset.

### C. Experiment 3: Effect of feature density

In order to investigate the effect of feature density on the performance of the MSCKF and SWF, we modified the "Starry Night" dataset by creating synthetic feature distributions with larger spatial extents and more features than the original dataset. We constructed each dataset so that the larger maps contained the same features as the smaller maps, plus additional features to make up the difference. Although the camera measurements in these datasets are synthetic, the IMU data are unchanged from the original dataset. Figure 8 shows the feature distribution for the 100-feature synthetic dataset. By generating longer feature tracks and increasing the number of features visible at each time step, these synthetic datasets allowed for a clearer comparison between the MSCKF and SWF.

Figure 9 compares translational and rotational Root Mean Squared Error (RMSE) of the MSCKF, SWF, and pure IMU integration on three synthetic datasets with 40, 60, and 100 features. With additional features, both algorithms consistently outperform pure IMU integration, and the SWF outperforms the MSCKF by a wide margin on all three datasets.

Note that the MSCKF's performance improves as the number of features increases, while the SWF's performance is not significantly affected by increasing feature count. This result indicates that the MSCKF is much more sensitive to feature density than the SWF. This may be due to the fact that the MSCKF updates its state whenever a feature goes

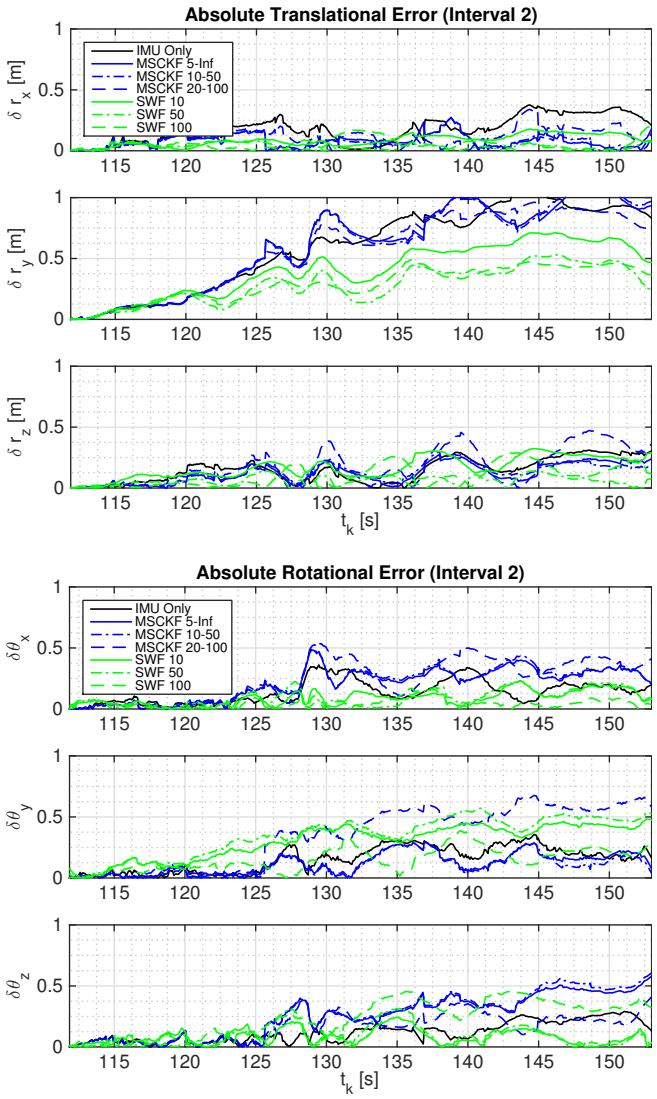


Fig. 7. Comparison of MSCKF, SWF, and IMU integration for multiple parameter settings on the dataset interval $k \in [1215, 1715]$. The numbers next to "MSCKF" in the legend refer to the minimum and maximum feature track lengths before an EKF update is triggered, while the number next to "SWF" refers to the number of states in the sliding window.

out of view and does not associate tracks corresponding to the same feature. If observations of a given feature are frequently interrupted, each track will not constrain the full set of poses from which the feature is visible. As feature density increases, the number of long feature tracks is likely to increase, and so the MSCKF estimates benefit from more high-quality motion constraints.

Since the SWF always associates all observations of each feature in a given window, it is not sensitive to the contiguity of the feature observations. However, [EXPLANATION ABOUT WHY IT DOESN'T IMPROVE MUCH WITH MORE FEATURES]

Table I provides summary statistics of each algorithm's performance on the synthetic datasets. As expected from Figure 9, the SWF outperforms the MSCKF in terms of average RMSE on both rotation estimation and position

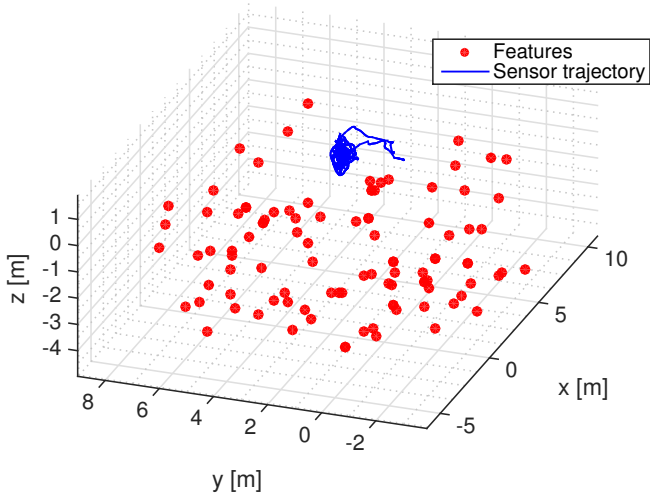


Fig. 8. Modified "Starry Night" dataset with 100 synthetic features. The sensor trajectory is identical to the original dataset.

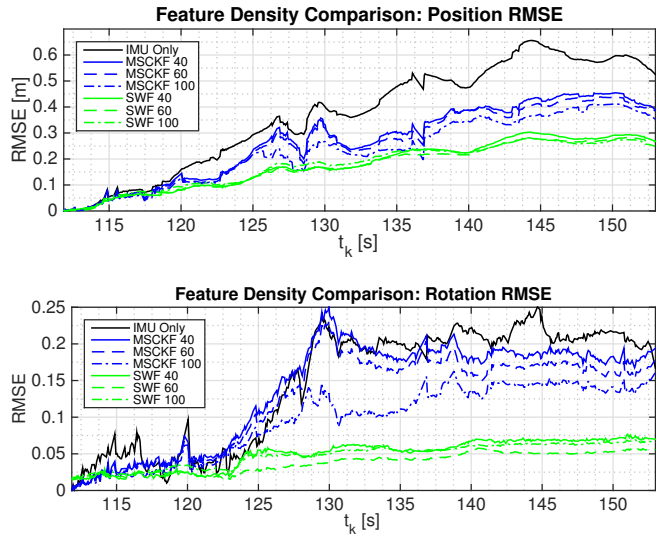


Fig. 9. Comparison of Root Mean Squared Error (RMSE) of the MSCKF, SWF, and pure IMU integration for three different synthetic maps generated based on the IMU data in the dataset interval $k \in [1215, 1715]$. The numbers next to "MSCKF" and "SWF" refer to the number of features in the dataset (40, 60 or 100).

estimation. It is worth noting, however, that, on average, the MSCKF achieves low Normalized Estimation Error Squared (NEES) values, which indicates that the filter scores well on consistency if not on accuracy. The average NEES values for the SWF are substantially higher because the SWF treats each window independently and has no built-in mechanism for propagating uncertainty from window to window ([4] describes the proper procedure using marginalization of old poses). The accumulated drift error in the SWF quickly escapes the estimator's reported uncertainty envelope, and so the SWF may be considered inconsistent in the absence of a known map.

Another point in favour of the MSCKF is that the computational effort required was significantly smaller than for the SWF. The MSCKF may therefore be better suited to vehicles with limited computational resources, particularly if they are operating in feature-rich environments.

TABLE I

COMPARISON OF AVERAGE ROOT MEAN SQUARED ERROR (ARMSE), AVERAGE NORMALIZED ESTIMATION ERROR SQUARED (ANEES), AND COMPUTE TIME OF IMU INTEGRATION, MSCKF, AND SWF ON SYNTHETIC DATASETS.

| | | Feature Count | | |
|---|---|---|---|---|
| | | 40 | 60 | 100 |
| IMU Only | Position ARMSE | 0.3679 | 0.3679 | 0.3679 |
| | Rotation ARMSE | 0.1452 | 0.1452 | 0.1452 |
| | ANEES | 0.2850 | 0.2850 | 0.2850 |
| | Compute Time† | 8.90 s | 8.90 s | 8.90 s |
| MSCKF | Position ARMSE | 0.2672 | 0.2550 | 0.2304 |
| | Rotation ARMSE | 0.1378 | 0.1247 | 0.0952 |
| | ANEES | 10.18 | 12.03 | 16.76 |
| | Compute Time† | 12.19 s | 14.64 s | 20.58 s |
| SWF | Position ARMSE | 0.1750 | 0.1687 | 0.1755 |
| | Rotation ARMSE | 0.0495 | 0.0377 | 0.0481 |
| | ANEES | 2280 | 2093 | 2013 |
| | Compute Time† | 114.3 s | 175.9 s | 245.3 s |

† Running MATLAB 2014b on a MacBook Pro with Intel i7 processor and 16 GB of RAM.

## VI. CONCLUSIONS

Based on our experimental results, we conclude that neither the SWF nor the MSCKF is more effective than pure IMU integration in environments with few features. In environments with more features, both algorithms provide substantial gains over pure IMU integration, although the SWF significantly outperformed the MSCKF in terms of accuracy on the datasets we tested. However, the accuracy of the MSCKF improves as more features are added, while the SWF is less sensitive to feature quantity. It is unclear from these results which algorithm would perform best as the number of features grows very large.

Although the SWF appears to produce more accurate pose estimates than the MSCKF in many cases, the MSCKF has the advantage of being less computationally intensive than the SWF, and of having better consistency properties. The MSCKF may be a better choice of algorithm in situations where computational resources are limited, or where a consistent estimate of state uncertainty is desirable.

## REFERENCES

[1] A. I. Mourikis, "Tech Note - A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation," pp. 1–20, Sep. 2006.

[2] A. I. Mourikis and S. I. Roumeliotis, "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation," in *Robotics and Automation, 2007 IEEE International Conference on*, 2007, pp. 3565–3572.

[3] A. I. Mourikis, M. Li, and B. H. Kim, "Real-time motion tracking on a cellphone using inertial sensing and a rolling-shutter camera," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 4712–4719.

[4] G. Sibley, G. Sukhatme, and L. Matthies, "Sliding window filter with application to planetary landing," *Journal of Field Robotics*, 2010.