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

Lee Clement, Valentin Peretroukhin, Jacob Lambert, and Jonathan Kelly

***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 estimate 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 characterizations and comparisons of the MSCKF and SWF on data from a moving hand-held sensor rig, as well as several traverses from the KITTI dataset. 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 linear 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 rapid motions. In these cases, IMU data can be relied upon more heavily in estimating egomotion.

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. This problem is often complicated by 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.

All authors are at the Institute for Aerospace Studies, University of Toronto, Canada {lee.clement, v.peretroukhin, jacob.lambert} @mail.utoronto.ca, jkelly@utias.utoronto.ca

Fig. 1. The sensor head used in our experiments. An IMU measures translational and rotational velocities, while a 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 characterize, compare, and contrast the performance of two modern solutions to the visual-inertial SLAM problem, the Sliding Window Filter (SWF) and the Multi-State Constraint Kalman Filter (MSCKF) [1], [2], on data from a moving hand-held sensor rig, as well as several traverses from the KITTI dataset [3].

II. Related Work

Visual-inertial navigation systems (VINS) have been applied broadly in robotics [2], [4]–[9], and there is a considerable body of work covering a wide range of estimation algorithms for the camera-IMU sensor pair. These techniques are often characterized as either *loosely coupled*, when image and IMU measurements are processed individually before being fused into a single estimate, or *tightly coupled*, when all information is processed together. The decoupling of inertial and visual measurements in loosely coupled systems leads to inexpensive computations, but at the cost of information: processing camera and IMU measurements separately makes optimal estimation of biases impossible [8]. In this work, we consider tightly coupled algorithms, which are preferable for accurate and consistent visual-inertial navigation.

For both tightly coupled and loosely coupled VINS, a popular choice of estimator is the Extended Kalman filter (EKF) or one of its variants [2], [5], [7], [9], [10], though methods also exist that employ unscented Kalman filters [11], particle filters [12], [13], or batch optimization methods [14], [15]. EKF-SLAM is an efficient recursive algorithm for small-scale, online tasks, but maintaining the entire map

as part of the state is a weakness for navigation over long distances. Indeed, the computational complexity of such algorithm scales far too poorly with the number of features to be applied naively to this problem. Additionally, EKF-based VINS are fundamentally inconsistent, that is, the state uncertainties are underestimated. Li and Mourikis [8] showed that the Jacobians in the linearized model of a VINS have different observability properties than the actual nonlinear system, severely reducing the reliability of such estimators.

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, EKF-SLAM sub-optimally estimates both vehicle motion and feature positions.

The shortcomings of the EKF motivate the use of filters that operate on a subset of the problem. Algorithms operating in constant time with respect to map size are particularly desirable. The Variable State Dimension Filter (VSDF) [16] is an early example of such algorithms. It applied a Kalman filter over a sliding window of states to achieve linear complexity with the number of features, taking advantage of *delayed linearization* in attempt to minimize linearization errors. The SWF analyzed in this work is similar in that it is both a recursive and batch approach, but unlike the VSDF, it incorporates a motion model from interoceptive data and applies batch optimization. The SWF operates in constant time by limiting the optimization problem to a set number of states at all times. Batch optimization over a window of states is suboptimal but still leads to exceptional accuracy for its cost, making it useful for delicate, large scale operations such as planetary landing [4]. The MSCKF [1], [2] can be thought of as a hybrid of EKF-SLAM and the SWF in the sense that it maintains a variable window of poses and applies batch updates using all observations of each landmark. It is also notable for its computationally efficiency, achieving accurate, real-time position tracking onboard a consumer smartphone [17].

III. Sliding Window Filter

The aim of the Sliding Window Filter (SWF) is to estimate a vehicle's motion by optimizing a sliding window of vehicle poses and observed landmarks. The optimization problem in the SWF is typically solved as a non-linear least squares problem using Gauss-Newton (GN) or Levenberg-Marquardt (LM) optimization over a sliding window of K poses $\mathbf{x}_{k\in[1,K]}$ observing M features $f_{j\in[1,M]}$:

$$\mathbf{x} := \begin{bmatrix}\mathbf{x}_0^T & \dots & \mathbf{x}_K^T & \mathbf{p}_G^{f_1 G\ T} & \dots & \mathbf{p}_G^{f_M G\ T}\end{bmatrix}^T. \quad (1)$$

In our notation, $\mathbf{p}_A^{BA}$ is the vector from the origin of frame $\mathcal{F}_A$ to the origin of frame $\mathcal{F}_B$ expressed in $\mathcal{F}_A$, and $\mathbf{C}_{BA}$ is the rotation matrix from $\mathcal{F}_A$ to $\mathcal{F}_B$.

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

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

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

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

where

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

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

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

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

with

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

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

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

$$\mathbf{T} := \text{diag}\left\{\mathbf{T}_1, \dots, \mathbf{T}_K\right\}, \quad (10)$$

where

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

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. 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 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 a subset of measurements of each feature contributing to the optimization. As a result, the filter may not maximally constrain some 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 variable 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 length of a feature track exceeds some preset threshold.

A. MSCKF State Parametrization

We evaluated the MSCKF with datasets 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 [1] and [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:

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

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

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

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

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

where $\otimes$ denotes quaternion multiplication.

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 $\hat{\mathbf{q}}_{IG,k}$.

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

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

and augment the MSCKF state covariance according to

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

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:

$$\dot{\hat{\mathbf{q}}}_{IG} = \frac{1}{2}\boldsymbol{\Omega}\left(\hat{\boldsymbol{\omega}}\right)\hat{\mathbf{q}}_{IG} \qquad \dot{\hat{\mathbf{b}}}_{\boldsymbol{\omega}} = \mathbf{0}_{3\times 1}$$

$$\dot{\hat{\mathbf{b}}}_{\mathbf{v}} = \mathbf{0}_{3\times 1} \qquad \dot{\hat{\mathbf{p}}}_G^{IG} = \hat{\mathbf{C}}_{IG}^T \hat{\mathbf{v}} \tag{21}$$

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:

$$\dot{\widetilde{\mathbf{x}}}_I = \mathbf{F}\widetilde{\mathbf{x}}_I + \mathbf{G}\mathbf{n}_I \tag{22}$$

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

$$\mathbf{F} = \begin{bmatrix} -\hat{\boldsymbol{\omega}}^\times & -\mathbf{1}_3 & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ -\hat{\mathbf{C}}_{IG}^T\hat{\mathbf{v}}^\times & \mathbf{0}_{3\times 3} & -\hat{\mathbf{C}}_{IG}^T & \mathbf{0}_{3\times 3} \end{bmatrix} \tag{23}$$

$$\mathbf{G} = \begin{bmatrix} -\mathbf{1}_3 & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{1}_3 & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{1}_3 \\ \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & -\hat{\mathbf{C}}_{IG}^T & \mathbf{0}_{3\times 3} \end{bmatrix}, \tag{24}$$

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 state covariances and IMU-camera cross-correlations as follows:

$$\hat{\mathbf{P}}^{-}_{CC,k+1} = \hat{\mathbf{P}}_{CC,k} \tag{25}$$

$$\hat{\mathbf{P}}^{-}_{IC,k+1} = \mathbf{\Phi}\left(t_k + T, t_k\right)\hat{\mathbf{P}}_{IC,k} \tag{26}$$

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

$$\dot{\mathbf{\Phi}}\left(t_k + \tau, t_k\right) = \mathbf{F}\mathbf{\Phi}\left(t_k + \tau, t_k\right), \tau \in [0, T] \tag{27}$$

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

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

$$\begin{aligned}\dot{\hat{\mathbf{P}}}_{II}\left(t_k + \tau\right) = \mathbf{F}\hat{\mathbf{P}}_{II}\left(t_k + \tau\right) + \hat{\mathbf{P}}_{II}\left(t_k + \tau\right)\mathbf{F}^T \\ + \mathbf{G}\mathbf{Q}_I\mathbf{G}^T, \tau \in [0, T]\end{aligned} \tag{28}$$

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

E. Feature Position Estimation

When a tracked feature f_j is selected for a state update, 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:

$$\hat{\mathbf{z}}_i^{(j)} = \begin{bmatrix} u_i' & v_i' \end{bmatrix}^T = \begin{bmatrix} (u_i - c_u)/f_u & (v_i - c_v)/f_v \end{bmatrix}^T. \tag{29}$$

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:

$$\hat{\mathbf{p}}_{C_1}^{f_j C_1} := \begin{bmatrix} \hat{X}_{C_1}^{(j)} & \hat{Y}_{C_1}^{(j)} & \hat{Z}_{C_1}^{(j)} \end{bmatrix}^T = \lambda \boldsymbol{\rho}_{C_1}^{f_j} \tag{30}$$

where

$$\boldsymbol{\rho}_{C_i}^{f_j} := \frac{1}{\sqrt{u_i'^2 + v_i'^2 + 1}} \begin{bmatrix} u_i' & v_i' & 1 \end{bmatrix}^T \tag{31}$$

is the direction of the ray emanating from camera C_i along which feature f_j must lie, and

$$\lambda = \left[\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\hat{\mathbf{p}}_{C_1}^{C_2 C_1} \right]_1 \tag{32}$$

with

$$\mathbf{A} := \begin{bmatrix} \boldsymbol{\rho}_{C_1}^{f_j} & -\boldsymbol{\rho}_{C_2}^{f_j} \end{bmatrix}. \tag{33}$$

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

$$\hat{\mathbf{p}}_{C_i}^{f_j C_i} = \hat{\mathbf{C}}_{i1}\hat{\mathbf{p}}_{C_1}^{f_j C_1} + \hat{\mathbf{p}}_{C_i}^{C_1 C_i}. \tag{34}$$

By forming the vector

$$\hat{\mathbf{y}} := \begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix}^T := \frac{1}{\hat{Z}_{C_1}^{(j)}} \begin{bmatrix} \hat{X}_{C_1}^{(j)} & \hat{Y}_{C_1}^{(j)} & 1 \end{bmatrix}^T, \tag{35}$$

we can define three functions

$$\begin{bmatrix} h_1(\hat{\mathbf{y}}) \\ h_2(\hat{\mathbf{y}}) \\ h_3(\hat{\mathbf{y}}) \end{bmatrix} = \hat{\mathbf{C}}_{i1} \begin{bmatrix} \alpha \\ \beta \\ 1 \end{bmatrix} + \gamma\hat{\mathbf{p}}_{C_i}^{C_1 C_i} \tag{36}$$

and rewrite the camera measurement error as

$$\mathbf{e}(\hat{\mathbf{y}}) = \hat{\mathbf{z}}_i^{(j)} - \frac{1}{h_3(\hat{\mathbf{y}})} \begin{bmatrix} h_1(\hat{\mathbf{y}}) \\ h_2(\hat{\mathbf{y}}) \end{bmatrix}. \tag{37}$$

The least-squares system then becomes

$$\left(\mathbf{E}^T\mathbf{W}^{-1}\mathbf{E}\right)\delta\mathbf{y}^* = -\mathbf{E}^T\mathbf{W}^{-1}\mathbf{e}(\hat{\mathbf{y}}) \tag{38}$$

where

$$\mathbf{E} = \frac{\partial \mathbf{e}}{\partial \hat{\mathbf{y}}} \quad \text{and} \quad \mathbf{W} = \text{diag}\left\{\mathbf{R}_1^{(j)}, \ldots, \mathbf{R}_N^{(j)}\right\} \tag{39}$$

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

F. MSCKF State Correction

Now that we have estimated the positions of any features to be used in the state update, 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 camera pose C_i:

$$\mathbf{r}_i^{(j)} := \mathbf{z}_i^{(j)} - \hat{\mathbf{z}}_i^{(j)} \tag{40}$$

where

$$\hat{\mathbf{z}}_i^{(j)} = \frac{1}{\hat{Z}_{C_i}^{(j)}} \begin{bmatrix} \hat{X}_{C_i}^{(j)} & \hat{Y}_{C_i}^{(j)} \end{bmatrix}^T \tag{41}$$

with

$$\begin{aligned}\hat{\mathbf{p}}_{C_i}^{f_j C_i} &= \begin{bmatrix} \hat{X}_{C_i}^{(j)} & \hat{Y}_{C_i}^{(j)} & \hat{Z}_{C_i}^{(j)} \end{bmatrix}^T \\ &= \hat{\mathbf{C}}_{C_i G}\left(\hat{\mathbf{p}}_G^{f_j G} - \hat{\mathbf{p}}_G^{C_i G}\right).\end{aligned} \tag{42}$$

If we linearize (40) about the estimates for the camera pose and feature position, we obtain an estimate of the exteroceptive measurement error

$$\mathbf{r}_i^{(j)} \simeq \mathbf{H}_{\mathbf{x},i}^{(j)}\tilde{\mathbf{x}}_i + \mathbf{H}_{f,i}^{(j)}\tilde{\mathbf{p}}_G^{f_j G} + \mathbf{n}_i^{(j)} \tag{43}$$

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 C_i with respect to the filter state and the position of the feature, respectively:

$$\mathbf{H}_{\mathbf{x},i}^{(j)} = \begin{bmatrix} \mathbf{0} & \mathbf{J}_i^{(j)}\left(\hat{\mathbf{p}}_{C_i}^{f_j C_i}\right)^{\times} & -\mathbf{J}_i^{(j)}\hat{\mathbf{C}}_{C_i G} & \mathbf{0} \end{bmatrix} \tag{44}$$

$$\mathbf{H}_{f,i}^{(j)} = \mathbf{J}_i^{(j)}\hat{\mathbf{C}}_{C_i G} \tag{45}$$

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

$$\mathbf{J}_i^{(j)} = \frac{1}{\left(\hat{Z}_{C_i}^{(j)}\right)^2} \begin{bmatrix} \hat{Z}_{C_i}^{(j)} & 0 & -\hat{X}_{C_i}^{(j)} \\ 0 & \hat{Z}_{C_i}^{(j)} & -\hat{Y}_{C_i}^{(j)} \end{bmatrix}. \tag{46}$$

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

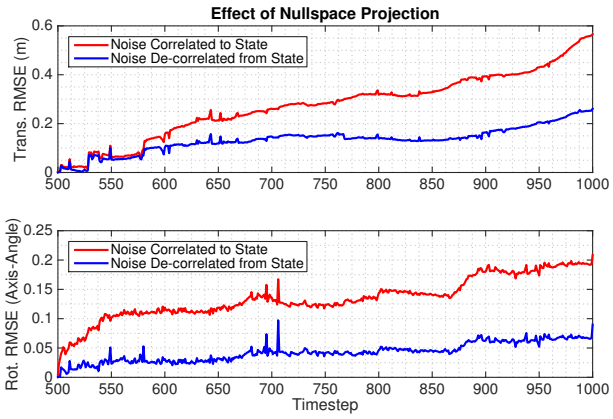


Fig. 2. Effect of correlation between measurement noise and filter state on MSCKF performance. The correlation between the measurement noise and the state causes the filter estimates to drift further from the true values than with a modified, de-correlated measurement noise.

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:

$$\mathbf{r}^{(j)} = \mathbf{z}^{(j)} - \hat{\mathbf{z}}^{(j)} \simeq \mathbf{H}_{\mathbf{x}}^{(j)}\tilde{\mathbf{x}} + \mathbf{H}_f^{(j)}\tilde{\mathbf{p}}_G^{f_j G} + \mathbf{n}^{(j)} \tag{47}$$

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. 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. As an aside, we note that the SWF does not have this problem because feature positions are jointly optimized along with the poses.

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):

$$\begin{aligned}\mathbf{r}_o^{(j)} &:= \mathbf{A}^T\mathbf{r}^{(j)} \simeq \mathbf{A}^T\mathbf{H}_{\mathbf{x}}^{(j)}\tilde{\mathbf{x}} + \mathbf{0} + \mathbf{A}^T\mathbf{n}^{(j)} \\ &=: \mathbf{H}_o^{(j)}\tilde{\mathbf{x}} + \mathbf{n}_o^{(j)}\end{aligned} \tag{48}$$

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

$$\mathbf{r}_o = \mathbf{H}_o\tilde{\mathbf{x}} + \mathbf{n}_o. \tag{49}$$

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:

$$\mathbf{H}_o = \begin{bmatrix}\mathbf{Q}_1 & \mathbf{Q}_2\end{bmatrix}\begin{bmatrix}\mathbf{T_H} \\ \mathbf{0}\end{bmatrix} \tag{50}$$

where $\mathbf{Q}_1, \mathbf{Q}_2$ are unitary matrices and $\mathbf{T_H}$ is an upper-triangular matrix. Substituting this result into (49) and pre-multiplying by $\begin{bmatrix}\mathbf{Q}_1 & \mathbf{Q}_2\end{bmatrix}^T$, we obtain

$$\begin{bmatrix}\mathbf{Q}_1^T\mathbf{r}_o \\ \mathbf{Q}_2^T\mathbf{r}_o\end{bmatrix} = \begin{bmatrix}\mathbf{T_H} \\ \mathbf{0}\end{bmatrix}\tilde{\mathbf{x}} + \begin{bmatrix}\mathbf{Q}_1^T\mathbf{n}_o \\ \mathbf{Q}_2^T\mathbf{n}_o\end{bmatrix}. \tag{51}$$

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:

$$\mathbf{r}_n := \mathbf{Q}_1^T\mathbf{r}_o = \mathbf{T_H}\tilde{\mathbf{x}} + \mathbf{Q}_1^T\mathbf{n}_o =: \mathbf{T_H}\tilde{\mathbf{x}} + \mathbf{n}_n \tag{52}$$

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:

$$\mathbf{K} = \hat{\mathbf{P}}_{k+1}^-\mathbf{T}_H^T\left(\mathbf{T}_H\hat{\mathbf{P}}_{k+1}^-\mathbf{T}_H^T + \mathbf{R}_n\right)^{-1} \tag{53}$$

$$\Delta\mathbf{x}_{k+1} = \mathbf{K}\mathbf{r}_n \tag{54}$$

$$\begin{aligned}\hat{\mathbf{P}}_{k+1} = &\left(\mathbf{1}_{12+6N} - \mathbf{K}\mathbf{T}_H\right)\hat{\mathbf{P}}_{k+1}^-\left(\mathbf{1}_{12+6N} - \mathbf{K}\mathbf{T}_H\right)^T \\ &+ \mathbf{K}\mathbf{R}_n\mathbf{K}^T.\end{aligned} \tag{55}$$

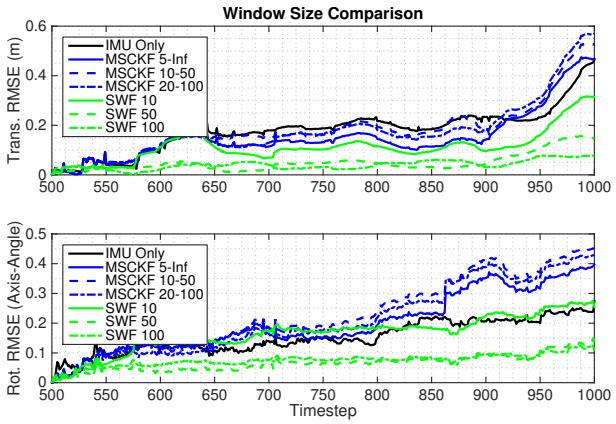
V. Experiments

We compared the performance of the MSCKF and SWF on two intervals in the "Starry Night" dataset from the University of Toronto Institute for Aerospace Studies (UTIAS), as well as 530 m of urban driving from the KITTI Dataset [3]. Both datasets include data streams from a stereo camera and IMU. Since our algorithms are designed to make use of a monocular camera, we artificially blinded the stereo camera by using images from the left camera only.

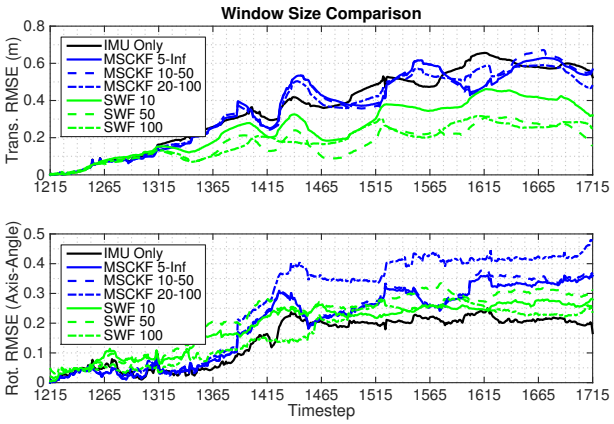
We implemented and tested both algorithms in MATLAB 2014b on a MacBook Pro Retina (11,3) with a 2.3GHz Intel Core i7 processor and 16 GB of DDR3L RAM. For both algorithms, we bootstrapped the filter window with one ground truth pose. For the SWF, we initialized feature positions using two-view triangulation.

A. Starry Night Dataset

The "Starry Night" 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 dataset is well-suited to evaluating SLAM algorithms since ground truth from a Vicon motion capture system is available for both the sensor head motion and the feature positions. We conducted three experiments on this dataset to compare the effect of feature visibility and window size on each algorithm. We discuss each of these in turn.



(a) At least three features are visible in 64% of the interval $(500, 1000)$. The number of visible features is frequently exceeds ten.



(b) At least three features are visible in only 56% of the interval $(1215, 1715)$. The number of visible features rarely exceeds ten.

Fig. 3. Comparison of MSCKF, SWF, and IMU integration for multiple parameter settings on two intervals of the "Starry Night" dataset. 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 numbers next to "SWF" refers to the number of states in the sliding window.

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

Figure 3(a) shows translational and rotational root mean squared errors (RMSE) for pure IMU integration and various parameter settings for the MSCKF and SWF. For the MSCKF, we varied the maximum number of observations before triggering an 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 window. These parameters did not significantly affect the accuracy of either filter, however Figure 3(a) shows some small gains in accuracy for larger feature track lengths and window sizes.

On this interval, the SWF outperforms the MSCKF in terms of both translational and rotational RMSE. 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.

2) Fewer visible features: In the second experiment, we compared the SWF and MSCKF for various parameter settings on an interval with few visible features. 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 3(b) shows translational and rotational RMSE 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.

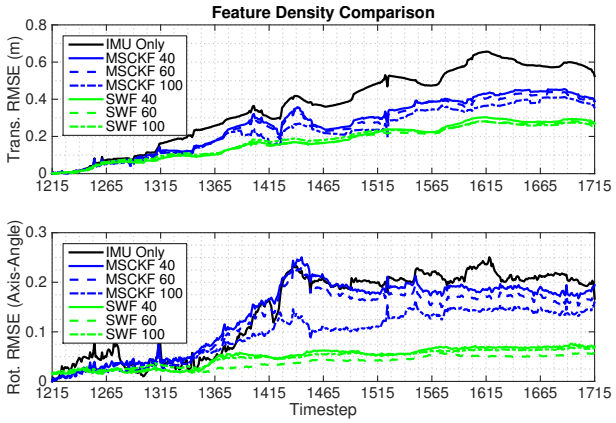


Fig. 4. Comparison of Root Mean Squared Error (RMSE) of the MSCKF, SWF, and pure IMU integration for three synthetic maps generated using the IMU data in the interval $(1215, 1715)$. The numbers next to "MSCKF" and "SWF" refer to the number of features in the dataset (40, 60 or 100).

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. In each dataset, we retained the IMU data from the original dataset and corrupted the synthetic camera measurements with zero-mean Gaussian noise. 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 4 compares translational and rotational RMSE of the MSCKF, SWF, and pure IMU integration on three synthetic datasets with 40, 60, and 100 features. With more 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 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 benefits 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.

Table I summarizes each algorithm's performance on the interval shown in Figure 4. As expected, the SWF outperforms the MSCKF in terms of average RMSE on both rotation and translation. 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 how this can be remedied by marginalizing old poses, a technique which was not implemented in this work). The accumulated drift error in the SWF quickly escapes the estimator's reported uncertainty envelope, and so this form of 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 an order of magnitude 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 FOR THE INTERVAL $(1215, 1715)$.

		Feature Count		
		40	60	100
IMU Only	Trans. ARMSE	0.3679	0.3679	0.3679
	Rot. 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 (20-100)	Trans. ARMSE	0.2672	0.2550	0.2304
	Rot. 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 (25)	Trans. ARMSE	0.1750	0.1687	0.1755
	Rot. 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 Retina (11,3) with a 2.3GHz Intel Core i7 processor and 16 GB of DDR3L RAM.

TABLE II

COMPARISON OF AVERAGE ROOT MEAN SQUARED ERROR (ARMSE) AND AVERAGE NORMALIZED ESTIMATION ERROR SQUARED (ANEES) IMU INTEGRATION, MSCKF, AND SWF ON KITTI TRAVERSES.

		KITTI Traverse			
		0001	0036	0051	0095
IMU Only	Trans. ARMSE	0.7197	0.5131	0.7834	1.039
	Rot. ARMSE	0.0025	0.0113	0.0034	0.0046
	ANEES	0.1630	0.0092	0.1170	0.6254
MSCKF (5-Inf)	Trans. ARMSE	0.3492	0.4401	0.7530	0.8170
	Rot. ARMSE	0.0111	0.0096	0.0091	0.0345
	ANEES	5.103	1.826	2.031	14.98
SWF (10)	Trans. ARMSE	0.3372	0.3778	0.5832	0.7196
	Rot. ARMSE	0.0113	0.0112	0.0111	0.0154
	ANEES	358.3	703.2	1124	3767

B. KITTI Dataset

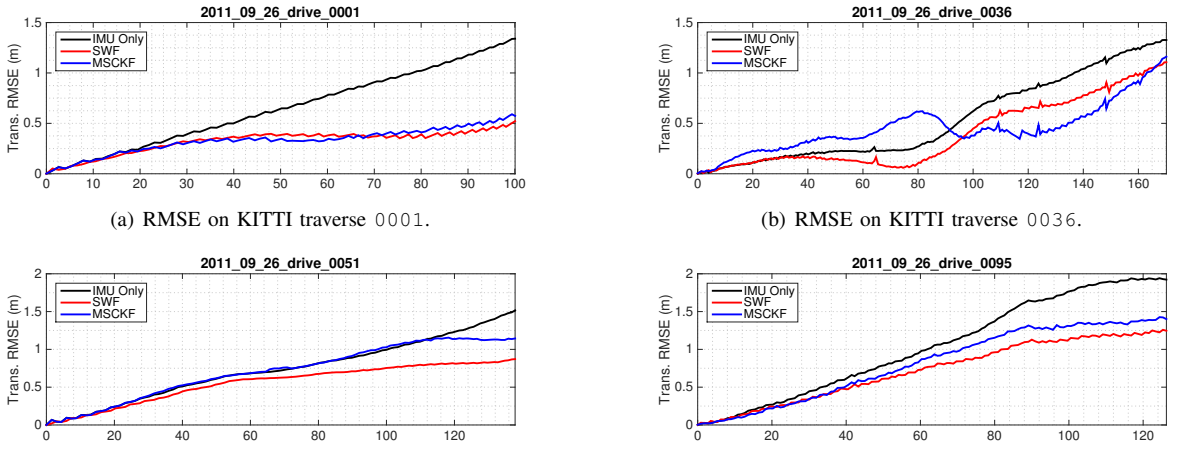
In addition to the "Starry Night" datasets, we also compared the performance of the MSCKF and SWF over several traverses from the KITTI dataset [3] totalling 530 m of urban driving. To accommodate our alternative state parametrization, we used the pre-integrated linear velocity measurements provided in the dataset instead of the raw linear accelerations from the IMU. Figure 5 shows the RMSE of pure IMU integration, the MSCKF, and the SWF over each of the four traverses we tested, representative images from which are shown in Figure 6. Similarly to the "Starry Night" results, both algorithms outperformed pure IMU integration, and the SWF slightly outperformed MSCKF in terms of rotational and translational RMSE. As expected, the MSCKF produced estimates that were more consistent than those of the SWF.

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. We stress that our inertial data was obtained from high-quality IMUs and has been sanitized to account for gravity, biases, and integration error. Using raw data from a consumer-grade IMU, we expect that pure IMU integration would have performed much worse, and that the benefit of the SWF and MSCKF would have been more apparent.

In environments with more features, both algorithms provided substantial gains in accuracy over pure IMU integration, with the SWF slightly outperforming the MSCKF 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 is less computationally intensive than the SWF and of has better consistency properties. The MSCKF may be a better choice of algorithm in applications where computational resources



(a) RMSE on KITTI traverse `0001`.

(b) RMSE on KITTI traverse `0036`.

(c) RMSE on KITTI traverse `0051`.

(d) RMSE on KITTI traverse `0095`.

Fig. 5. Translational and rotational RMSE over four traverses from the KITTI dataset [3] totaling 534 m of urban driving.

Fig. 6. Sample frames from the four KITTI traverses shown in Figure 5.

are limited and the operational environment is feature-rich.

REFERENCES

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

[2] A. I. Mourikis and S. I. Roumeliotis, "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2007, pp. 3565–3572.

[3] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[4] G. Sibley, L. Matthies, and G. Sukhatme, "Sliding window filter with application to planetary landing," *J. Field Robotics*, vol. 27, no. 5, pp. 587–608, 2010.

[5] A. Huster, E. Frew, and S. Rock, "Relative position estimation for auvs by fusing bearing and inertial rate sensor measurements," in *Proc. OCEANS*, vol. 3, 2002.

[6] D. G. Kottas, K. J. Wu, and S. I. Roumeliotis, "Detecting and dealing with hovering maneuvers in vision-aided inertial navigation systems."

[7] J. Kima and S. Sukkariehb, "Real-time implementation of airborne inertial-slam," in *Proc. Robotics: Science and Systems (RSS)*, 2007.

[8] M. Li and A. I. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *Int. J. Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

[9] S. You and U. Neumann, "Fusion of vision and gyro tracking for robust augmented reality registration."

[10] D. G. Kottas and S. I. Roumeliotis, "Exploiting urban scenes for vision-aided inertial navigation," in *Proc. Robotics: Science and Systems (RSS)*, 2013.

[11] S. Ebcin and M. Veth, "Tightly-coupled image-aided inertial navigation using the unscented kalman filter," in *Proc. Int. Tech. Meeting of the Satellite Division of The Institute of Navigation*, 2007, pp. 1851–1860.

[12] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *J. Artificial Intelligence Research*, vol. 11, pp. 391–427, 1999.

[13] M. Pupilli and A. Calway, "Real-time camera tracking using a particle filter," in *Proc. British Mach. Vis. Conf. (BMVC)*, 2005, pp. 519–528.

[14] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment – a modern synthesis," in *Vision Algorithms: Theory and Practice*. Springer Verlag, 2000, pp. 298–375.

[15] D. Strelow and S. Singh, "Motion estimation from image and inertial measurements," *Int. J. Robotics Research*, vol. 23, no. 12, pp. 1157 – 1195, December 2004.

[16] P. F. McLauchlan and D. W. Murray, "Variable state dimension filter applied to active camera calibration," vol. 2059, 1993, pp. 14–25.

[17] 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 *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2013, pp. 4712–4719.