

# An Improved Multi-State Constraint Kalman Filter for Visual-Inertial Odometry

M.R. Abdollahi<sup>†,a</sup>, Seid H. Pourtakdousti<sup>a</sup>, M.H. Yoosefian Nooshabadi<sup>†,b</sup>, H.N. Pishkenari<sup>b</sup>

<sup>a</sup>*Department of Aerospace Engineering, Sharif University of Technology, Tehran, Iran*

<sup>b</sup>*Department of Mechanical Engineering, Sharif University of Technology, Tehran, Iran*

---

## Abstract

Fast pose estimation (PE) plays a vital role for agile autonomous robots in successfully carrying out their tasks. While Global Navigation Satellite Systems (GNSS) such as Global Positioning System (GPS) have been traditionally used along with Inertial Navigation Systems (INS) for PE, their viability is compromised in indoor and urban environments due to their low update rates and inadequate signal coverage. Visual-Inertial Odometry (VIO) is gaining popularity as a practical alternative to GNSS/INS systems in GNSS-denied environments. Among various VIO-based methods, the Multi-State Constraint Kalman Filter (MSCKF) has garnered significant attention due to its robustness, speed and accuracy. Nevertheless, high computational cost of image processing is still challenging for real-time implementation on resource-constrained vehicles. In this paper, an enhanced version of the MSCKF is proposed. The proposed approach differs from the original MSCKF in the feature marginalization and state pruning steps of the algorithm. This new design results in a faster algorithm with comparable accuracy. We validate the proposed algorithm using both an open-source dataset and real-world experiments. It is demonstrated that the proposed Fast-MSCKF (referred to as FMSCKF) is approximately six times faster and at least 20% more accurate in final position estimation compared to the standard MSCKF.

**Keywords:** MSCKF, Fast MSCKF, Visual-Inertial Odometry, Agile Motion, Kalman Filter.

---

[video link](#): experimental test

<sup>†</sup>These authors contributed equally to this work.

## 1. Introduction

In recent years, the utilization of quadrotors has witnessed a significant upsurge across various applications (Abbasi et al., 2018b,a; Abdollahi et al., 2019). Accurate and reliable pose estimation serves as a crucial step in designing guidance and control systems for quadrotors. GPS and IMU data fusion has been widely used for pose estimation in autonomous vehicles (Sukkarieh et al., 1999; Cen et al., 2020; Macias et al., 2021; Cheng et al., 2022). However, this method has some drawbacks in certain scenarios. For instance, satellite signals may not be readily available in various settings, such as forests, indoor areas, and urban environments with tall buildings. Moreover, the update rate of many satellite receivers is notably low and with a slight delay. As a result, alternative approaches have been explored to replace GNSS/INS systems and address their limitations. Although it does not provide absolute pose (i.e., the pose of the system with respect to an Earth-fixed frame), camera-IMU fusion is one of the emerging methods for relative PE, due to the low cost and high quality of the sensors. This approach is commonly referred to as Visual Inertial Odometry (VIO) in the relevant literature. VIO approaches are considered safe, cost-effective and robust replacements for GNSS/INS systems in applications where it suffices to have relative PE.

There are various methods for VIO in the literature, including ORB-SLAM (Mur-Artal et al., 2015), SVO+GTSAM (Forster et al., 2016a), CNN-SVO (Loo et al., 2019), DSO (Engel et al., 2017), VINS-Mono (Qin et al., 2018), and Kimera (Rosinol et al., 2020). VIO methods can be categorized into two major groups, namely, *filter-based* and *optimization-based* approaches. Optimization-based methods tend to be more computationally expensive, which makes them suitable for offline and post-processing applications. On the other hand, the low computational load and relatively high accuracy of filter-based methods renders them well-suited for real-time implementations (Huang, 2019). Given that this paper’s focus is on fast PE, we opt for the latter approach. In certain filter-based methods, landmark positions are stored in the state vector (see ROVIO (Bloesch et al., 2017), EKF- and UKF- SLAM (Brossard et al., 2018)). However, as the number of landmark positions increases, the size of the state vector grows proportionally, resulting in a considerable rise in computational demands (Delmerico and Scaramuzza, 2018). Therefore, for real-time applications, only a limited number of landmark positions can be used. On the other hand, some other filter-based approaches, such as the MSCKF, store a number of camera poses in the state vector.

The Multi-State Constraint Kalman Filter (MSCKF) was originally developed by Mourikis and Roumeliotis in their seminal paper (Mourikis and Roumeliotis, 2007). The MSCKF uses an Error-State Extended Kalman filter to fuse IMU and camera data and, unlike other KF-based methods, does not store the positions of landmarks

in the state vector. Instead, a series of recent positions and orientations (i.e., poses) of the camera are kept in the state vector. This alternative strategy significantly reduces the computational cost while enhancing estimation accuracy and robustness.

In general, to improve the performance of a VIO algorithm, three main characteristics need to be considered, namely, *accuracy*, *robustness*, and *output rate*. There are many studies in the literature that attempt to improve these three features. To improve accuracy, a Patch-based MSCKF (Zheng et al., 2017), has been developed. Unlike the original MSCKF algorithm, in which image features are utilized, in the patch-based MSCKF, a direct method is employed. In this method, the light intensities of pixels in distinct patches are used for estimation. This approach achieves, on average, 23% enhancement in accuracy compared to the original MSCKF (Zheng et al., 2017).

The Local-Optimal MSCKF (LOMSCKF) has been developed in an attempt to improve estimation accuracy using nonlinear optimization (Forster et al., 2016a; Heo et al., 2018). The LOMSCKF employs pre-integrated IMU and camera measurements. Moreover, Stereo MSCKF (S-MSCKF), which uses a stereo camera instead of a single camera, has been proposed to enhance the performance of the original MSCKF. The S-MSCKF has shown more robustness with a modest increase in computational cost (Sun et al., 2018).

One limitation of EKF-based methods lies in the inherent unobservability of the yaw angle, rendering them intrinsically inconsistent. Huang has considered this problem and proposed potential solutions, including Observability-Constrained EKF (OC-EKF) (Hesch et al., 2013) and First-Estimates Jacobian (FEJ-EKF) (Huang et al., 2009). Furthermore, MA has introduced Ackermann MSCKF (ACK-MSCKF) algorithm (Ma et al., 2019), in which the MSCKF has been modified to be used for ground vehicles. In this work, the unobservable states of ground robots are investigated, and an algorithm is presented to resolve the unobservability problem.

There are also numerous recent works focusing on the integration of machine learning techniques into visual-inertial navigation to enhance robustness and accuracy (Chen et al., 2020; Li and Waslander, 2020). For instance, a convolutional neural network has been implemented in the update step of the MSCKF algorithm, which has resulted in higher robustness compared to the original MSCKF (Zuo et al., 2021). However, learning-based approaches often lack formal convergence guarantees, which makes them unreliable for safety-critical application.

As discussed above, numerous studies have been conducted with the goal of enhancing the original MSCKF algorithm. The majority of these endeavors have aimed to improve the algorithm’s accuracy or address the observability issue associated with the yaw angle. However, relatively few studies have tried to decrease the computational cost of the algorithm. Yet, the high computational cost remains the primary challenge for resource-constrained robots. Therefore, reducing this computational

cost is of great importance. To this end, this paper attempts to reduce the computational cost of the MSCKF algorithm using a new feature management method.

The rest of the paper is organized as follows. The next section covers the formulation of the MSCKF. Section 3 constitutes the core contribution of this work, elucidating the feature management methodology of both the original MSCKF and the proposed FMSCKF. The implementation results obtained from both the MSCKF and FMSCKF on an open-source dataset and in real-world experiments are presented in section 4. In section 5, a detailed discussion of the results is provided. Finally, conclusions are provided in section 6.

## 2. The Multi State Constraint Kalman Filter (MSCKF)

The MSCKF is an error-state extended Kalman filter that can be used to estimate the position and orientation of a camera-IMU system. More precisely, the algorithm estimates the pose of the frame attached to the IMU,  $\{I\}$ , with respect to a global (reference) frame,  $\{G\}$ . The equations in this section are based on (Mourikis and Roumeliotis, 2007). The state vector of the filter consists of two parts. The first part, which evolves during the propagation step, is the IMU state

$$\mathbf{X}_I = [{}^I \mathbf{q}_G^\top \quad \mathbf{b}_g^\top \quad {}^G \mathbf{v}_I^\top \quad \mathbf{b}_a^\top \quad {}^G \mathbf{p}_I^\top]^\top, \quad (1)$$

in which  ${}^I \mathbf{q}_G \in \mathbb{R}^4$  is the unit quaternion indicating the rotation from the frame  $\{G\}$  to the frame  $\{I\}$ ,  ${}^G \mathbf{v}_I \in \mathbb{R}^3$  and  ${}^G \mathbf{p}_I \in \mathbb{R}^3$  are the velocity and position vectors of the frame  $\{I\}$  with respect to and expressed in the frame  $\{G\}$ , and  $\mathbf{b}_g \in \mathbb{R}^3$  and  $\mathbf{b}_a \in \mathbb{R}^3$  are gyroscope and accelerometer biases, respectively. The second part of the state vector includes a number of poses of the camera frames  $\{C\}$  with respect to the global frame  $\{G\}$ . This part is added to the state vector during the augmentation step (section 2.2). The algorithm consists of three steps, which are explained in the following sections.

### 2.1. The Propagation Step

Every time a new IMU measurement is received, the first part of the state vector and its corresponding covariance matrix are propagated. Considering a calibrated IMU (i.e., misalignment, temperature effects, scale-factor, etc. are accounted for), the measurement models for the gyroscope and the accelerometer are

$$\boldsymbol{\omega}_m = \boldsymbol{\omega} + \mathbf{b}_g + \mathbf{n}_g, \quad (2)$$

$$\mathbf{a}_m = {}^I \mathbf{R}_G ({}^G \mathbf{a} - {}^G \mathbf{g}) + \mathbf{b}_a + \mathbf{n}_a, \quad (3)$$

in which  $\boldsymbol{\omega}_m \in \mathbb{R}^3$  and  $\boldsymbol{\omega} \in \mathbb{R}^3$  are the measured and true angular velocities,  $\mathbf{a}_m \in \mathbb{R}^3$  is the measured body acceleration,  ${}^I \mathbf{R}_G \in \mathbb{R}^{3 \times 3}$  is the rotation matrix corresponding

to the quaternion  ${}^I\mathbf{q}_G$ ,  $\mathbf{n}_g \in \mathbb{R}^3$  and  $\mathbf{n}_a \in \mathbb{R}^3$  are zero mean white process noise vectors,  ${}^G\mathbf{a} \in \mathbb{R}^3$  is the acceleration of the frame  $\{I\}$  expressed in  $\{G\}$ , and finally,  ${}^G\mathbf{g} \in \mathbb{R}^3$  is the local gravity vector expressed in  $\{G\}$ . Bias vectors are modeled as random walk processes, hence, the time derivatives of entries of the IMU state vector is

$${}^I\dot{\mathbf{q}}_G(t) = \frac{1}{2}\boldsymbol{\Omega}({}^I\boldsymbol{\omega}(t)) {}^I\mathbf{q}_G(t), \quad (4)$$

$${}^G\dot{\mathbf{p}}_I(t) = {}^G\mathbf{v}_I(t), \quad (5)$$

$${}^G\dot{\mathbf{v}}_I(t) = {}^G\mathbf{a}_I(t), \quad (6)$$

$$\dot{\mathbf{b}}_g(t) = \mathbf{n}_{b_g}(t), \quad (7)$$

$$\dot{\mathbf{b}}_a(t) = \mathbf{n}_{b_a}(t). \quad (8)$$

In (7) and (8),  $\mathbf{n}_{b_g} \in \mathbb{R}^3$  and  $\mathbf{n}_{b_a} \in \mathbb{R}^3$  are random walk noise vectors and

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} -[\boldsymbol{\omega} \times] & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^\top & \mathbf{0} \end{bmatrix}, \quad [\boldsymbol{\omega} \times] = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ \omega_y & \omega_x & 0 \end{bmatrix}. \quad (9)$$

Applying the expectation operator to equations (4) to (8) yields

$${}^I\dot{\hat{\mathbf{q}}}_G = \frac{1}{2}\boldsymbol{\Omega}(\hat{\boldsymbol{\omega}}) {}^I\hat{\mathbf{q}}_G, \quad (10)$$

$${}^G\dot{\hat{\mathbf{p}}}_I = {}^G\hat{\mathbf{v}}_I, \quad (11)$$

$${}^G\dot{\hat{\mathbf{v}}}_I = {}^I\mathbf{R}_G^\top \hat{\mathbf{a}} + {}^G\mathbf{g}, \quad (12)$$

$$\dot{\hat{\mathbf{b}}}_g = \mathbf{0}, \quad (13)$$

$$\dot{\hat{\mathbf{b}}}_a = \mathbf{0}, \quad (14)$$

where  $\hat{\boldsymbol{\omega}} = \boldsymbol{\omega}_m - \hat{\mathbf{b}}_g$  and  $\hat{\mathbf{a}} = \mathbf{a}_m - \hat{\mathbf{b}}_a$ . These equations can be solved either by continuous-time integration (using differential equations) or discrete-time integration (using methods such as fourth order Runge-Kutta) to obtain the IMU state propagation equations. The covariance matrix is also propagated. This matrix, at time step  $k+1$ , can be partitioned as

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{II_{k+1|k}} & \mathbf{P}_{IC_{k+1|k}} \\ \mathbf{P}_{CI_{k+1|k}} & \mathbf{P}_{CC_{k|k}} \end{bmatrix}, \quad (15)$$

in which the  $\mathbf{P}_{II_{k+1|k}}$  is the covariance matrix related to the IMU state (the correlation between IMU state variables) and can be calculated through numerical integration of the following Lyapunov equation

$$\dot{\mathbf{P}}_{II} = \mathbf{F}\mathbf{P}_{II} + \mathbf{P}_{II}\mathbf{F}^\top + \mathbf{G}\mathbf{Q}_I\mathbf{G}^\top, \quad (16)$$

in which

$$\mathbf{F} = \begin{bmatrix} -[\hat{\boldsymbol{\omega}} \times] & -\mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ -\hat{\mathbf{R}}_I [\hat{\mathbf{a}} \times] & \mathbf{0}_3 & \mathbf{0}_3 & -\hat{\mathbf{R}}_I & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix}, \quad (17)$$

$$\mathbf{G} = \begin{bmatrix} -\mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & -\mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & -\hat{\mathbf{R}}_I & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & -\mathbf{I}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix}, \quad (18)$$

where  $\hat{\mathbf{R}}_I$  and  $[\hat{\mathbf{a}} \times]$  are clear from the context, and  $\mathbf{Q}_I$  is the covariance matrix corresponding to the noise vector

$$\mathbf{n}_I = [\mathbf{n}_g^\top \quad \mathbf{n}_{b_g}^\top \quad \mathbf{n}_a^\top \quad \mathbf{n}_{b_a}^\top]^\top. \quad (19)$$

The next term in the covariance matrix in (15) is  $\mathbf{P}_{IC_{k+1|k}}$ , which represents the correlation between the IMU and the camera states and is propagated using

$$\mathbf{P}_{IC_{k+1|k}} = \Phi(t_k + T, t_k) \mathbf{P}_{IC_{k|k}}, \quad (20)$$

where  $T$  is the sampling time of the IMU, and  $\Phi$  is the state transition matrix that evolves over time according to the following equation

$$\dot{\Phi}(t_k + \tau, t_k) = \mathbf{F}\Phi(t_k + \tau, t_k), \quad (21)$$

with  $\mathbf{F}$  given in (17).

## 2.2. The Augmentation Step

Upon recording an image, the camera pose is calculated based on the most recent estimate of the IMU pose. Assume that the filter is currently at the  $(k + 1)$ -th time

step, and the state vector already includes  $l$  camera poses. When the  $(l + 1)$ -th image is received, its pose is calculated using

$${}^{C_{l+1}}\hat{\mathbf{q}}_G = {}^C\mathbf{q}_I \otimes {}^{I_{k+1}}\hat{\mathbf{q}}_G, \quad (22)$$

$${}^G\hat{\mathbf{p}}_{C_{l+1}} = {}^G\hat{\mathbf{p}}_{I_{k+1}} + {}^G\hat{\mathbf{R}}_{I_{k+1}} {}^I\mathbf{p}_C, \quad (23)$$

in which  ${}^I\mathbf{p}_C$  and  ${}^C\mathbf{q}_I$  are the translation vector and the quaternion between the IMU frame and the camera frame, respectively. Since the two sensors are rigidly attached to each other,  ${}^I\mathbf{p}_C$  and  ${}^C\mathbf{q}_I$  are two constants, and can be computed offline. In the next step, the state vector is augmented according to

$$\hat{\mathbf{X}}_{aug} = \begin{bmatrix} \hat{\mathbf{X}}^\top & {}^{C_{l+1}}\hat{\mathbf{q}}_G^\top & {}^G\hat{\mathbf{p}}_{C_{l+1}}^\top \end{bmatrix}^\top, \quad (24)$$

in which  $\hat{\mathbf{X}}$  is the state vector prior to augmentation. In addition to the state vector, the covariance matrix should be augmented using

$$\mathbf{P} = \begin{bmatrix} \mathbf{I}_{6l+15} \\ \mathbf{J} \end{bmatrix} \mathbf{P} \begin{bmatrix} \mathbf{I}_{6l+15} \\ \mathbf{J} \end{bmatrix}^\top, \quad (25)$$

in which  $\mathbf{J}$  is the Jacobian matrix, that is calculated using

$$\mathbf{J} = \begin{bmatrix} {}^C\mathbf{R}_I & \mathbf{0}_{3 \times 9} & \mathbf{0}_3 & \mathbf{0}_{3 \times 6l} \\ \left[ \left( {}^G\hat{\mathbf{R}}_{I_{k+1}} {}^I\mathbf{p}_C \right) \times \right] & \mathbf{0}_{3 \times 9} & \mathbf{I}_3 & \mathbf{0}_{3 \times 6l} \end{bmatrix}. \quad (26)$$

### 2.3. The Update Step

The update step is carried out using features extracted from images, the poses of which are available in the state vector. The policy based on which these features are chosen is the subject of section 3, and is the key distinction between the proposed method and the original MSCKF. Assume  $m$  features are observed in  $n_j$  images  $(l - n_j, \dots, l - 1)$  with  $l$  being the index of the most recent recorded image. Each image contains pixel coordinates corresponding to different features. Therefore, the camera measurement model for the  $j$ -th ( $j = 1, \dots, m$ ) feature observed in the  $i$ -th ( $i = 1, \dots, n_j$ ) image is

$${}^{C_i}\mathbf{z}_{f_j} = \frac{1}{{}^{C_i}Z_{f_j}} \begin{bmatrix} {}^{C_i}X_{f_j} \\ {}^{C_i}Y_{f_j} \\ {}^{C_i}Z_{f_j} \end{bmatrix} + {}^{C_i}\mathbf{n}_{f_j}, \quad (27)$$

where  ${}^{C_i}\mathbf{n}_{f_j} \in \mathbb{R}^2$  is the measurement noise with the covariance  ${}^{C_i}\mathbf{R}_{f_j} = \sigma_{im}^2 \mathbf{I}_2$ , and  ${}^{C_i}\mathbf{p}_{f_j} = [{}^{C_i}X_{f_j} \quad {}^{C_i}Y_{f_j} \quad {}^{C_i}Z_{f_j}]^\top$  is the position of the  $j$ -th feature in the  $i$ -th camera frame. Utilizing all  $n_j$  measurements of the feature, it is possible to estimate the

position of the feature in the  $\{G\}$  frame via triangulation (see (Triggs et al., 2000; Nousias et al., 2019)). Assume that the estimated position of the  $j$ -th feature in the global frame is denoted as  ${}^G\hat{\mathbf{p}}_{f_j}$ . The estimate of the position of the  $j$ -th feature in the  $i$ -th image frame is calculated via

$${}^{C_i}\hat{\mathbf{p}}_{f_j} = {}^{C_i}\hat{\mathbf{R}}_G ({}^G\hat{\mathbf{p}}_{f_j} - {}^G\hat{\mathbf{p}}_{C_i}) = \begin{bmatrix} {}^{C_i}\hat{X}_{f_j} \\ {}^{C_i}\hat{Y}_{f_j} \\ {}^{C_i}\hat{Z}_{f_j} \end{bmatrix}. \quad (28)$$

The residual is calculated using the measurement of the feature

$${}^{C_i}\mathbf{r}_{f_j} = {}^{C_i}\mathbf{z}_{f_j} - {}^{C_i}\hat{\mathbf{z}}_{f_j}, \quad (29)$$

in which

$${}^{C_i}\hat{\mathbf{z}}_{f_j} = \frac{1}{{}^{C_i}\hat{Z}_{f_j}} \begin{bmatrix} {}^{C_i}\hat{X}_{f_j} & {}^{C_i}\hat{Y}_{f_j} \end{bmatrix}^\top. \quad (30)$$

Subsequently, the Jacobians of the measurement model with respect to the position of the feature,  ${}^{C_i}\mathbf{H}_{f_j}$ , and with respect to the state vector,  ${}^{C_i}\mathbf{H}_{X_j}$ , are calculated using

$${}^{C_i}\mathbf{H}_{f_j} = {}^{C_i}\mathbf{J}_{f_j} {}^{C_i}\hat{\mathbf{R}}_G, \quad (31)$$

$${}^{C_i}\mathbf{H}_{X_j} = [\mathbf{0}_{2 \times 15} \ \dots \ {}^{C_i}\mathbf{H}_{f_j} [({}^G\hat{\mathbf{p}}_{f_j} - {}^G\hat{\mathbf{p}}_{C_i}) \times] \ - {}^{C_i}\mathbf{H}_{f_j}], \quad (32)$$

where

$${}^{C_i}\mathbf{J}_{f_j} = \left( \frac{1}{{}^{C_i}\hat{Z}_{f_j}} \right)^2 \begin{bmatrix} {}^{C_i}\hat{Z}_{f_j} & 0 & -{}^{C_i}\hat{X}_{f_j} \\ 0 & {}^{C_i}\hat{Z}_{f_j} & -{}^{C_i}\hat{Y}_{f_j} \end{bmatrix}. \quad (33)$$

These calculations are performed for all  $n_j$  images in which the  $j$ -th feature has been observed. Concatenating all calculations for the  $j$ -th feature yields

$$\mathbf{r}_{f_j} = \begin{bmatrix} {}^{C_1}\mathbf{r}_{f_j}^\top & \dots & {}^{C_{n_j}}\mathbf{r}_{f_j}^\top \end{bmatrix}^\top, \quad (34)$$

$$\mathbf{H}_{f_j} = \begin{bmatrix} {}^{C_1}\mathbf{H}_{f_j}^\top & \dots & {}^{C_{n_j}}\mathbf{H}_{f_j}^\top \end{bmatrix}^\top, \quad (35)$$

$$\mathbf{H}_{X_j} = \begin{bmatrix} {}^{C_1}\mathbf{H}_{X_j}^\top & \dots & {}^{C_{n_j}}\mathbf{H}_{X_j}^\top \end{bmatrix}^\top. \quad (36)$$

Let  $\mathbf{A}_{f_j}$  be the left null-space of  $\mathbf{H}_{f_j}$ . Define

$$\mathbf{H}_{o_j} = \mathbf{A}_{f_j}^\top \mathbf{H}_{X_j}, \quad (37)$$

$$\mathbf{R}_{o_j} = \sigma_{im}^2 \mathbf{A}_{f_j}^\top \mathbf{A}_{f_j}, \quad (38)$$

$$\mathbf{r}_{o_j} = \mathbf{A}_{f_j}^\top \mathbf{r}_{f_j}. \quad (39)$$

Matrices  $\mathbf{H}_{o_j}$ ,  $\mathbf{R}_{o_j}$ , and  $\mathbf{r}_{o_j}$  are calculated for all  $m$  features, which are to be used to carry out the update step. Stacking the calculated matrices for all  $m$  features yields

$$\mathbf{H}_o = [\mathbf{H}_{o_1}^\top \cdots \mathbf{H}_{o_m}^\top]^\top, \quad (40)$$

$$\mathbf{r}_o = [\mathbf{r}_{o_1}^\top \cdots \mathbf{r}_{o_m}^\top]^\top, \quad (41)$$

$$\mathbf{R}_o = \text{diag}(\mathbf{R}_{o_1}, \dots, \mathbf{R}_{o_m}), \quad (42)$$

where  $\mathbf{r}_o \in \mathbb{R}^d$  is the residual computed for all  $m$  features observed in  $n_j$  images, and

$$d = \sum_{j=1}^m (2n_j - 3).$$

Since the number of features can be large, an additional step can be taken to reduce the computational cost. Specifically, reduced QR-decomposition can be utilized to decompose the  $\mathbf{H}_o$  matrix as

$$\mathbf{H}_o = [\mathbf{Q}_1 \ \mathbf{Q}_2] \begin{bmatrix} \mathbf{T}_H \\ \mathbf{0} \end{bmatrix}, \quad (43)$$

where  $\mathbf{Q}_1$  and  $\mathbf{Q}_2$  are unitary matrices, and  $\mathbf{T}_H$  is an upper triangular matrix. As a result, the matrices calculated using (40) to (42) can be reduced to

$$\mathbf{H}_n = \mathbf{T}_H, \quad (44)$$

$$\mathbf{r}_n = \mathbf{Q}_1^\top \mathbf{r}_o, \quad (45)$$

$$\mathbf{R}_n = \mathbf{Q}_1^\top \mathbf{R}_o \mathbf{Q}_1. \quad (46)$$

Next, the Kalman gain is computed via

$$\mathbf{K} = \mathbf{P}_{k+1|k} \mathbf{H}_n^\top \mathbf{S}^{-1}, \quad (47)$$

where

$$\mathbf{S} = \mathbf{H}_n \mathbf{P}_{k+1|k} \mathbf{H}_n^\top. \quad (48)$$

Finally, the Kalman gain in (47) is used to update the covariance matrix and the state vector according to

$$\mathbf{P}_{k+1|k+1} = \mathbf{P}_{k+1|k} - \mathbf{K} \mathbf{S} \mathbf{K}^\top, \quad (49)$$

$$\tilde{\mathbf{X}} = \mathbf{K} \mathbf{r}_n, \quad (50)$$

$$\widehat{\mathbf{X}}_{k+1|k+1} = \widehat{\mathbf{X}}_{k+1|k} \oplus \widetilde{\mathbf{X}}, \quad (51)$$

where the  $\oplus$  represents the general form of addition, which for the quaternion entry of the state vector is equivalent to the quaternion multiplication, and for other entries is the same as normal addition. The updated state vector and covariance matrix will be used in the next propagation step of the subsequent iteration of the filter.

Note that since the MSCKF is an Error-State Extended Kalman Filter, the orientation error-state is minimal, as we define

$$\delta \mathbf{q} = \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\theta} \\ 1 \end{bmatrix},$$

where  $\delta\boldsymbol{\theta} \in \mathbb{R}^3$  represents the angular error. To enhance numerical stability and ensure that the quaternion norm constraint is met, we normalize the quaternion vector after both the propagation step and the update step. In addition to quaternion normalization, we make the covariance matrix symmetric. For more information, the reader is referred to (Sola, 2017, §5).

### 3. Feature Marginalization And State Pruning

In section 2, it was assumed that  $m$  number of features are used to perform the update step of the MSCKF algorithm. In this section, the approach used to select these features is explained. In the conventional MSCKF, the update step is carried out when one of the two following cases happen.

- *Features are no longer visible.*

This case, which happens most often, occurs when at least one feature is no longer visible in the new image. Each time an image is captured, some of the previously visible features may fall out of the field of view and therefore cannot be tracked in the new image. These features are then used to perform the update step in the algorithm. To maintain the number of features at a constant level, an equivalent number of new features are extracted in the latest image.

Over time, when all features from an image have been utilized in the update steps or are no longer trackable, that image no longer provides any useful information for the algorithm. At this point, the pose associated with that image is removed from the state vector. Additionally, the corresponding row and column related to this pose are pruned from the covariance matrix. This state and covariance pruning is essential as it prevents the state vector and the covariance matrix from growing indefinitely, which would otherwise lead to computational inefficiency and increased memory usage.

- *State vector reaches a certain size.*

This situation happens when the number of camera poses available in the state vector reaches its maximum,  $N_{p_{max}}$ . This case usually occurs when the camera is not moving fast enough. Consequently, when the camera moves slowly, although some features may fall outside the field of view and be used to carry out an update (previous case), the state vector and the covariance matrix are not pruned. This is due to the fact that the rate at which the features fall outside the field of view is not sufficiently high, so, the sizes of the state vector and the covariance matrix keep increasing. As a result, over time, the number of stored camera poses reaches its maximum,  $N_{p_{max}}$ . In this case, at least one image should be removed. Before removing that image, all its features and their tracks across preceding images are used to perform an update. This ensures that the information contained in those features is not lost. After carrying out the update, the image, along with its corresponding entries in the state vector and the covariance matrix, is removed. Instead of removing only one image, in the original MSCKF,  $N_{p_{max}}/3$  images are removed. Starting from the second-oldest pose, these  $N_{p_{max}}/3$  images are evenly spaced in time (Mourikis and Roumeliotis, 2007). The reason for keeping the oldest image is that the geometric constraint involving this image usually corresponds to a large baseline, making it carry more valuable information about positioning. According to (Mourikis and Roumeliotis, 2007), this method performs well in practice.

Hence, in the conventional MSCKF there is only one tuning parameter which is  $N_{p_{max}}$ .

On the other hand, within the Fast MSCKF (FMSCKF) framework, we introduce an additional scenario centered on the minimum number of features that are tracked. In this case, in contrast to the conventional MSCKF approach, where features are extracted from every recorded image, our FMSCKF methodology focuses solely on a subset of images, which we term as *keyframes*. More specifically, in this approach, once features are extracted from a *keyframe*, they are tracked in the subsequent frames, and no new features are extracted until the number of tracked features falls below a specific threshold, denoted as  $N_{f_{min}}$ . When the number of tracked features is less than the threshold, all feature tracks are used to perform an update. Post-update, all images, except the last image (the keyframe) are removed, and the state vector and covariance matrix are pruned accordingly. Following that, the final image in the sequence is utilized for the extraction of new features, which will be tracked in forthcoming images.

It is worth mentioning that the concept of keyframes used in this work differs significantly from the common keyframe-based methods found in the literature. Traditionally, keyframe selection is based on the change in the baseline. This means that when the positional change between two consecutive images is large enough, these

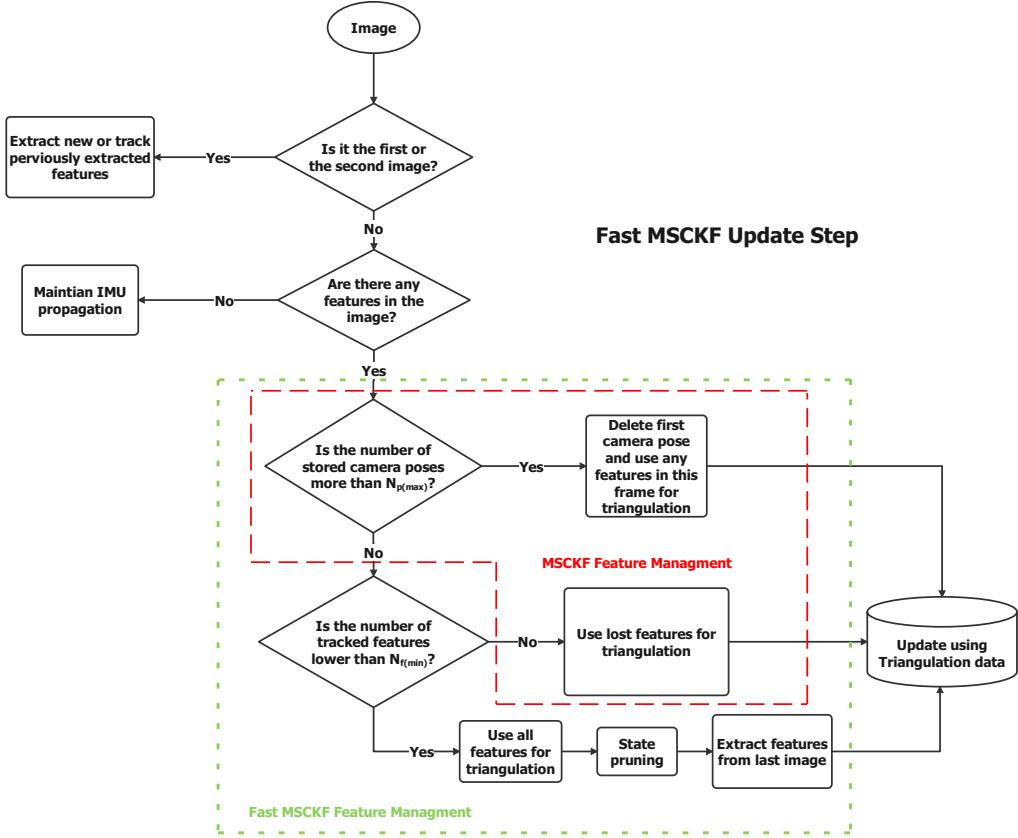


Figure 1: The flowchart of feature management approaches in the MSCKF and the FMSCKF algorithms.

images are designated as keyframes (Azimi et al., 2022). However, our work adopts a different criterion for defining keyframes. Instead of relying on the change in the baseline, we determine keyframes based on the trackability of features within the frames. Specifically, in our method, a frame  $l$  (where  $l > 1$ ) is considered a keyframe if the number of features that can be tracked in it falls below a certain threshold,  $N_{f_{min}}$ . This threshold is a predefined minimum number of features to be tracked, and when the number of these features in a frame drops below this threshold, that frame is marked as a keyframe.

Therefore, within the proposed strategy, three distinct scenarios can trigger an update in the filter, which are as follows in descending order of frequency, from the most frequent to the least frequent.

- *Features are no longer visible.*

The first case is similar to the first case in the conventional MSCKF and occurs when at least one feature exits the field of view of the camera. This case happens most frequently.

- *Feature track reaches a certain number.*

The second scenario, which happens second most often, occurs when the number of tracked features falls below a minimum threshold,  $N_{f_{min}}$ . As explained before, in this case, all feature tracks are used to perform an update. Then, all images in the sequence, except for the most recent one (i.e., the keyframe), are removed. This removal process involves pruning the state vector and the covariance matrix to maintain a manageable size and prevent computational inefficiencies. Following the pruning process, the final image in the sequence, is then used for the extraction of new features. These newly extracted features will be tracked in subsequent images.

- *State vector reaches a certain size.*

Lastly, the third case, which is equivalent to the second scenario in the conventional MSCKF, happens when the number of camera poses available in the state vector exceeds a maximum number,  $N_{p_{max}}$ . This case happens least often.

It is imperative to note that the introduced feature marginalization method has proven to be significantly faster and more accurate in practical applications than the original approach. A schematic of the two feature management methods is shown in Figure 1. The pseudocode for the update step of the FMSCKF is presented in Algorithm 1.

To provide a visual comparison between the two feature management policies, Figure 2 shows four consecutive frames and their corresponding features extracted in the MSCKF (right column) and the FMSCKF (left column). As can be seen from Figure 2, the number of tracked features in the first three frames of the FMSCKF algorithm is decreasing. In the fourth frame, this number falls below the threshold, and as a result, the algorithm extracts new features. On the other hand, the number of features tracked in the MSCKF algorithm remains constant throughout all four frames.

## 4. Results

In this section, the results of the proposed algorithm are presented. For the original MSCKF (Mourikis and Roumeliotis, 2007), the maximum allowable number of poses is chosen to be  $N_{p_{max}} = 20$ . For the FMSCKF, the minimum number of features is set to  $N_{f_{min}} = 8$ , and similar to the original MSCKF, the maximum allowable number of

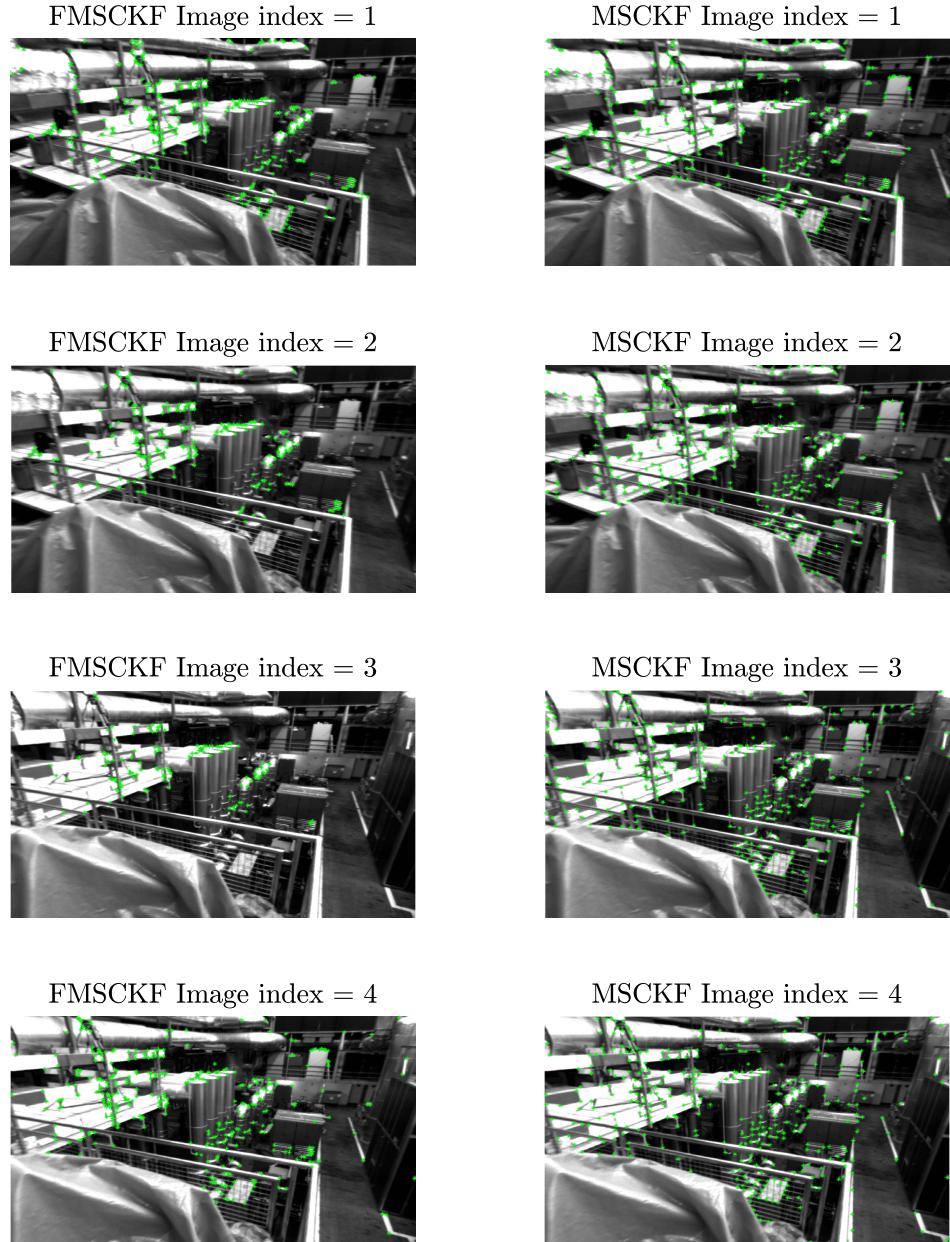


Figure 2: Comparison of the number of extracted features in the FMSCKF and the MSCKF. In the fourth frame on the left column, the number of tracked features falls below the threshold, and therefore, the algorithm extracts new features. However, the number of features in the original MSCKF algorithm, shown in the right column, is constant. The images are parts of MH\_01 dataset (Burri et al., 2016).

---

**Algorithm 1:** Pseudocode for the update step in the proposed algorithm.

---

**Input** : Augmented state vector, augmented covariance matrix,  $m$  features tracked in  $n_j$  images.

**Output:** Updated state vector and covariance matrix.

```
1 if number of stored camera poses <  $N_{p_{max}}$  then
2   for  $j = 1, \dots, m$  do
3     Calculate the position of the  $j$ -th feature in the global frame using triangulation;
4     for  $i = 1, \dots, n_j$  do
5       Calculate the position of the  $j$ -th feature in the  $i$ -th frame using (28);
6       Estimate the measurement using (30), then calculate the residual in (29);
7       Use (31), (32), and (33) to calculate  ${}^{c_i}\mathbf{H}_{f_j}$ ,  ${}^{c_i}\mathbf{H}_{X_j}$ , and  ${}^{c_i}\mathbf{J}_{f_j}$ ;
8     end
9     Use (34) to (36) to compute  $\mathbf{r}_{f_j}$ ,  $\mathbf{H}_{f_j}$ , and  $\mathbf{H}_{X_j}$ ;
10    Calculate the left null-space of  $\mathbf{H}_{f_j}$ ;
11    Calculate  $\mathbf{H}_{o_j}$ ,  $\mathbf{r}_{o_j}$ , and  $\mathbf{R}_{o_j}$  using (37) to (39);
12    Perform a Mahalanobis gating test to reject the outliers;
13  end
14  Calculate  $\mathbf{H}_o$ ,  $\mathbf{r}_o$ , and  $\mathbf{R}_o$  using (40) to (42);
15  Perform a QR-decomposition and calculate  $\mathbf{H}_n$ ,  $\mathbf{r}_n$ , and  $\mathbf{R}_n$ , using (44) to (46);
16  Update the covariance matrix and the state vector using (49) and (51);
17 else if number of tracked features <  $N_{f_{min}}$  then
18   For all features run 2 - 16;
19   Prune the state vector and the covariance matrix;
20   Extract new features from the last image;
21 else
22   Delete the first camera pose from the state vector, and its corresponding entries from the covariance matrix;
23   For all features in the deleted frame run 2 - 16;
24 end
```

---

Table 1: The update rate (in  $Hz$ ) for various algorithms implemented on the EuRoC MAV dataset (Burri et al., 2016). The codes were executed on MATLAB.

Dataset	ROVIO	SVO (VIO)	SVO+GTSAM	MSCKF	VINS_Mono	FMSCKF
MH_01	38.75	32.51	64.67	28.49	35.31	<b>106.38</b>
MH_02	41.98	35.13	70.06	30.86	36.12	<b>117.65</b>
MH_03	37.36	31.45	62.36	27.47	35.06	<b>93.46</b>
MH_04	44.01	37.1	73.46	32.36	35.31	<b>101.01</b>
MH_05	40.72	34.35	67.96	29.94	35.82	<b>114.94</b>
VR1_01	34.43	29.23	57.47	25.32	32.87	<b>84.75</b>
VR1_02	39.08	32.91	65.23	28.74	35.46	<b>107.53</b>
VR1_03	37.06	31.12	61.85	27.25	34.73	<b>99.01</b>
VR2_01	38.64	33.19	64.49	28.41	35.73	<b>111.11</b>
VR2_02	34.78	30.54	58.06	25.58	35.04	<b>96.15</b>
VR2_03	36.66	31.43	61.19	26.95	35.17	<b>103.09</b>

poses is chosen to be  $N_{p_{max}} = 20$ . The rationale for selecting  $N_{f_{min}} = 8$  is elaborated in Section 5. In the keyframes, a maximum of 350 Harris corners are extracted and tracked in consecutive frames. To track the extracted features, the well-known KLT algorithm (Tomasi and Kanade, 1993) is used. The Random Sample Consensus (RANSAC) algorithm is used to reject outliers in feature tracking (Fischler and Bolles, 1981).

#### 4.1. Public Dataset

The EuRoC MAV dataset (Burri et al., 2016) is used to evaluate the algorithm. To provide a comprehensive comparison of the update rate, the results of SVO (VIO) (Forster et al., 2016b), SVO+GTSAM (Forster et al., 2016a), VINS-mono (Qin et al., 2018) and ROVIO (Bloesch et al., 2017) algorithms are presented. Default settings are applied for these algorithms. All computations were conducted on a personal MacBook Air equipped with an M1 chip and 8GB Memory.

The update rates for different algorithms are presented in Table 1. The superior performance of the FMSCKF compared to other algorithms in terms of update rate is evident from Table 1. To visually compare the algorithms, the Root Mean Square Error (RMSE) plots in position and orientation estimation for one dataset (MH\_01) are shown in Figures 3 and 4, respectively. From Figure 3, it can be seen that the

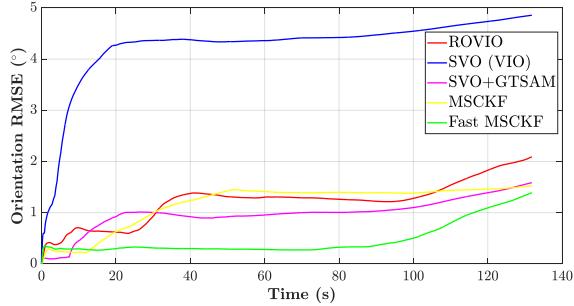


Figure 3: orientation estimation RMSE for different algorithms.

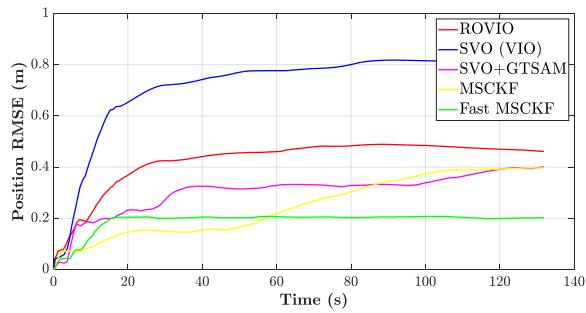


Figure 4: position estimation RMSE for different algorithms.

FMSCKF outperforms all the other algorithms in terms of orientation estimation. One should note that the ascending behavior of orientation error in Figure 3 is due to the fact that in EKF-based visual-inertial estimators, the yaw angle is not observable (Li and Mourikis, 2013) and therefore, it drifts over time. From Figure 4, it can be deduced that the FMSCKF has a lower error in position estimation compared to other algorithms. Additionally, in Figure 5, the 3D paths for one dataset (MH\_01) using the MSCKF and the FMSCKF, alongside the ground-truth are shown. Final errors in orientation and position estimation for these algorithms are provided in Table 2.

Frame processing time for MH\_01 is shown in Figure 6. As expected, the FMSCKF processes the frames significantly faster than the original MSCKF.

Moreover, box plots illustrating Absolute Trajectory Error (ATE) and Relative Trajectory Error (RTE) are depicted in Figures 7 and 8, respectively. As can be seen in Figures 7 and 8, the FMSCKF demonstrates accuracy on par with other algorithms. Notably, it frequently surpasses the MSCKF in both ATE and RTE, while concurrently achieving a significantly higher output rate, as shown in Table 1. Note that we did not include the ATE and RTE results of SVO+GTSAM for MH\_03 in Figures 7 and 8, as its errors were notably higher than compared to the other

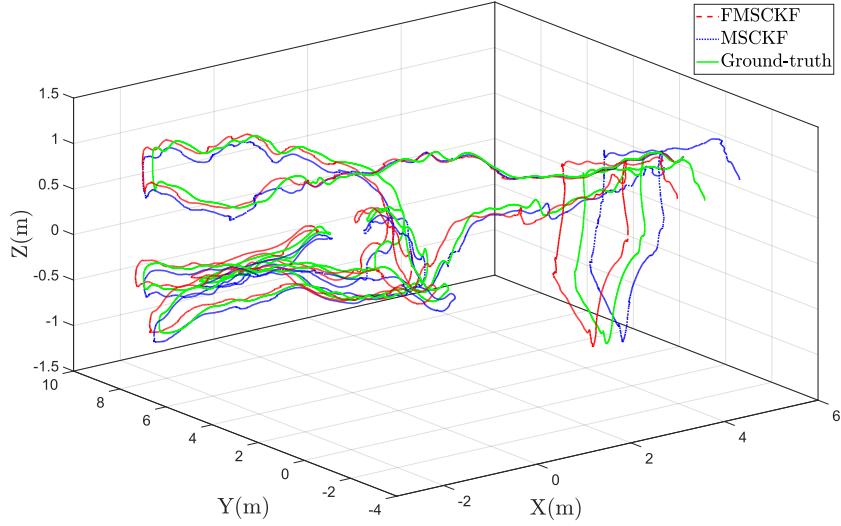


Figure 5: Estimated 3D paths for MH\_01 using the MSCKF (blue) and the FMSCKF (red). The ground-truth path is plotted in green (Burri et al., 2016).

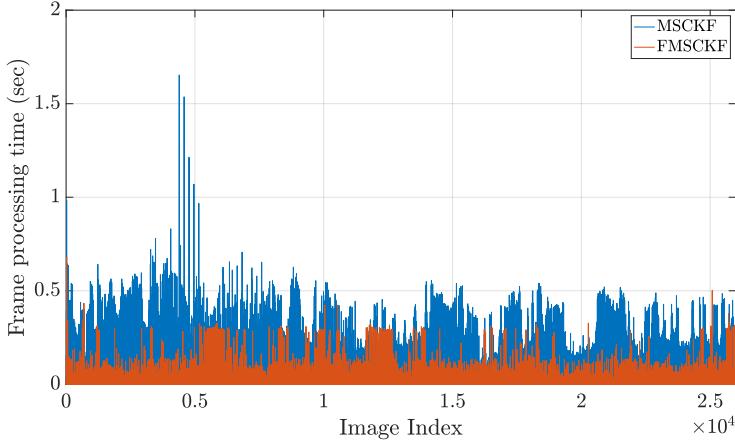


Figure 6: Frame processing time for the FMSCKF (red) and the MSCKF (blue).

algorithms.

Furthermore, the number of tracked features, camera poses, and features used for update are shown in Figures 9, 10, and 11, respectively. As evident, both the number of tracked features and the number of features used for update in the FMSCKF are lower than those in the MSCKF, which, as claimed, results in a higher update rate

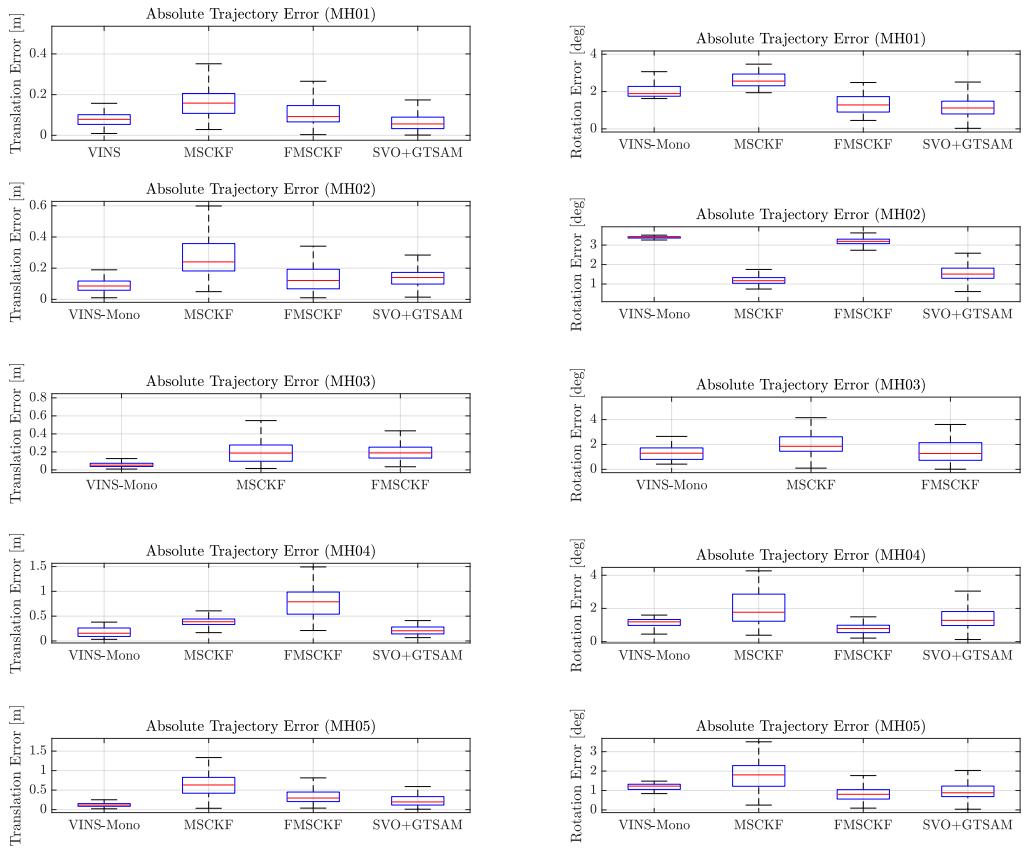


Figure 7: Absolute Trajectory Error (ATE) for various algorithms.

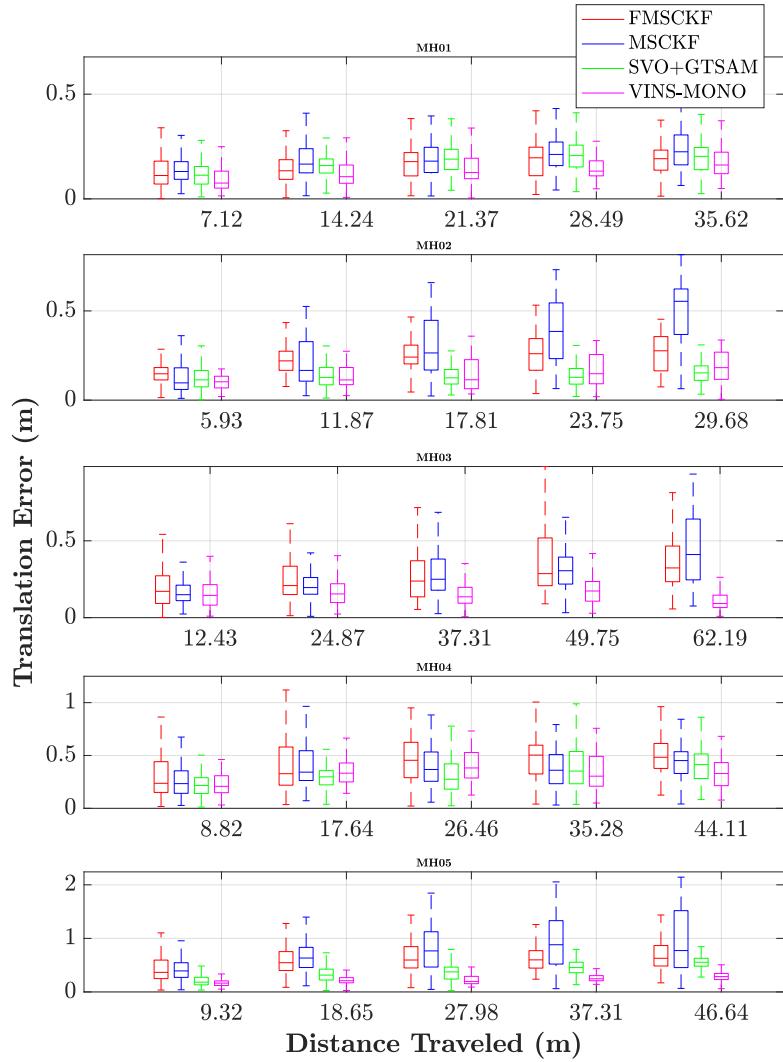


Figure 8: Relative Trajectory Error (RTE) for various algorithms.

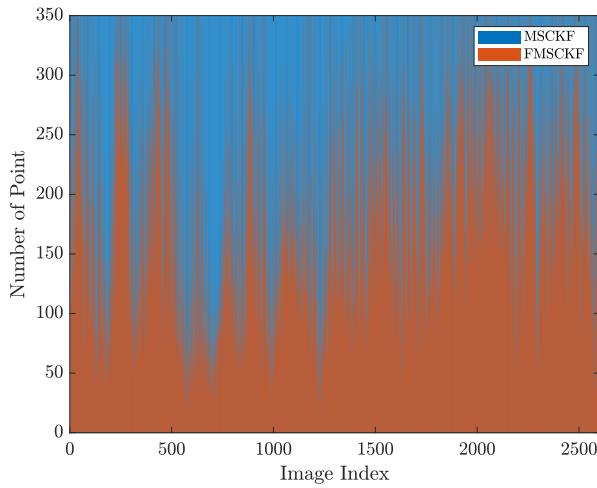


Figure 9: Number of tracked features in each image for the MSCKF (blue) and the FMSCKF (red).

of the FMSCKF.

#### 4.2. Real-world Experiments

An experimental setup was developed and utilized to test the proposed algorithm. The main processor utilized was an NVIDIA Jetson Xavier NX. The setup is shown in Figure 12. Further details regarding the sensors employed in the platform can be found in Table 1 3. The setup was calibrated using Kalibr library (Rehder et al., 2016). The algorithms were not executed in real-time; instead, the acquired data was later used on the personal laptop to run the filters. Three experiments were conducted, the results of which will be presented in the following subsections.

##### 4.2.1. Short-range experiment

In the first experiment, a path spanning 45 meters was traversed in approximately 90 seconds. A sample image recorded during this experiment is shown in Figure

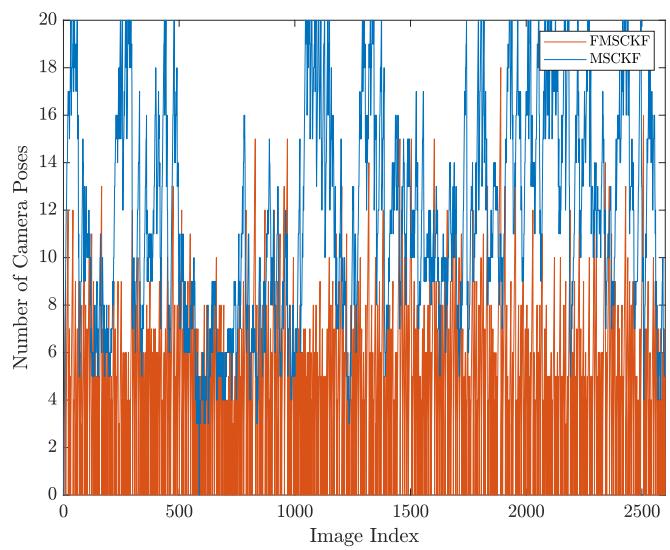


Figure 10: Number of camera poses stored in the state vector for the FMSCKF (blue) and the MSCKF (red).

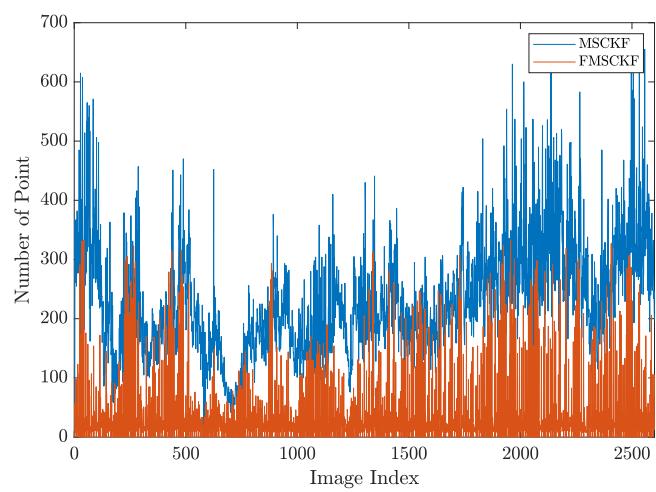


Figure 11: Number of features used in the update step of the filter the MSCKF (blue) and the FMSCKF (red).

Table 2: Results of various algorithms for MH\_01 (Burri et al., 2016). The final point error is calculated with respect to the traveled distance.

Method	Final point error (%)	Final orientation error ( $o$ )
ROVIO	0.57	2.09
SVO (VIO)	0.98	4.86
SVO+GTSAM	0.49	1.59
MSCKF	0.49	1.52
FMSCKF	<b>0.25</b>	<b>1.39</b>

Table 3: The sensor setup used in this paper.

Sensor	Type	Output rate (Hz)
ADIS 16467 – 2	MEMS IMU	100
Basler acA1300 – 200uc	Global shutter camera	10

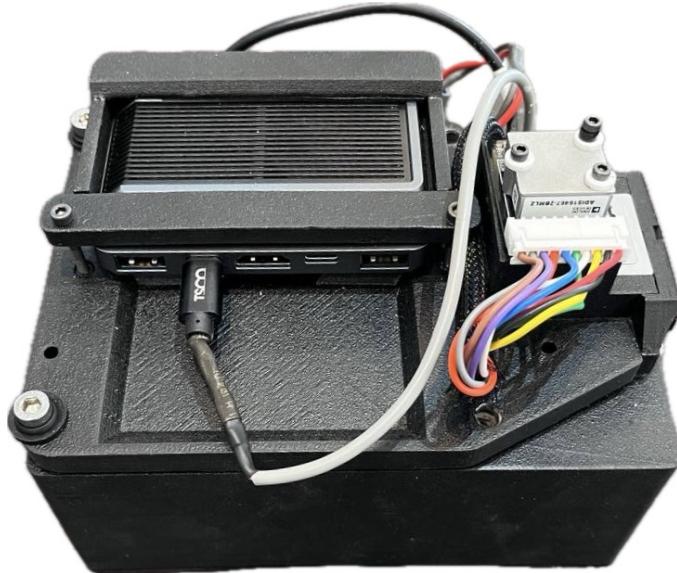


Figure 12: The customized sensor setup. The camera is mounted on the front side. An STM32 F103c8t6 receives data from the IMU, and after an initial processing, sends it to the Nvidia Jetson board.



Figure 13: A sample image of the short-range dataset.

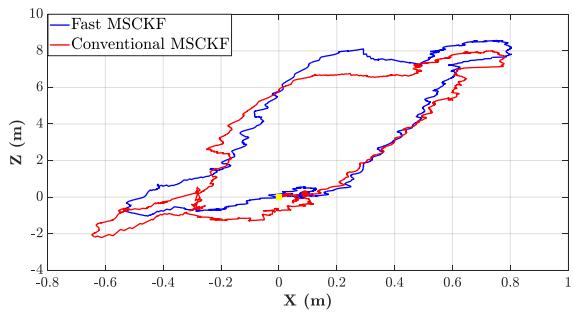


Figure 14: Estimated path in XZ plane using the FMSCKF (blue) and the MSCKF (red) in the short-range experiment. The starting point is indicated by a yellow square and final points are marked with circles.

13. For a more precise and explicit comparison, only the original MSCKF and the proposed FMSCKF were employed for this experiment.

The setup was moved by a person along a random path, making it impossible to recover the ground-truth trajectory. As a result, to evaluate the performance of the two algorithms, we attempted to create a loop-shaped path and return precisely to the starting point. This enabled us to measure and compare the final point error for each algorithm. The estimated trajectories are shown in Figure 14. As evident from Figure 14, both algorithms exhibit good performance in position estimation. It is worth mentioning that the difference in the estimated trajectories of the two algorithms is primarily due to the low quality of the camera.

The final position estimates of the MSCKF and the FMSCKF, as well as the ground truth, are listed in Table 4.

Accordingly, the final point error of the FMSCKF is  $0.14(m)$ , while that of the MSCKF is  $0.24(m)$ . In other words, for the 45-meter-long trajectory, the final point error of the FMSCKF is 0.31% and the final point error of the MSCKF is 0.53% of the traveled distance. These errors are in good compliance with the final point errors

Table 4: Results of the short-range experiment.

Algorithm	$x_f(m)$	$y_f(m)$	$z_f(m)$	Final point error (%)
FMSCKF	0.0806	0.0355	0.1085	0.31
MSCKF	0.0899	-0.1385	0.1747	0.53
Ground truth	0	0	0	N.A.

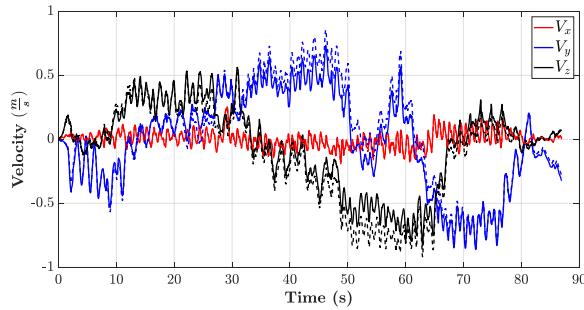


Figure 15: Estimated velocities during the short-range experiment using the FMSCKF (solid) and the MSCKF (dashed).

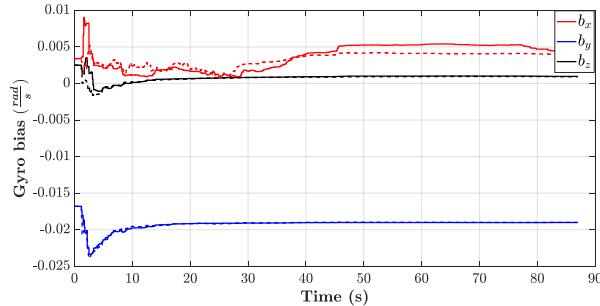


Figure 16: Estimated gyroscope biases during the short-range experiment using the FMSCKF (solid) and the MSCKF (dashed).

of the algorithms running on the EuRoC MAV dataset. For instance, for MH\_01, these numbers are 0.25% and 0.49%, respectively. Estimated velocities and biases are plotted in Figures 15 to 17.

#### 4.2.2. Mid-range experiment

In the second experiment, the setup was mounted on a car, which then traversed a 900-meter-long path. A sample image recorded during this experiment is shown

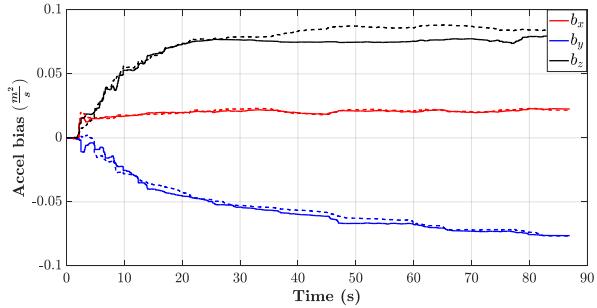


Figure 17: Estimated accelerometer biases during the short-range experiment using the FMSCKF (solid) and the MSCKF (dashed).



Figure 18: A sample image of the mid-range dataset.

in Figure 18. The estimated paths using the FMSCKF and the MSCKF along with the ground-truth path are shown in Figure 19. Note that, similar to the short-range test, GPS signal was inaccurate due to presence of tall buildings. However, unlike the short-range test, the ground-truth was obtained by reconstructing the trajectory of the vehicle based on the lanes it traveled in. It is worth noting that since this ground-truth reconstruction is prone to various errors, it shall not be used for numerical evaluation of the algorithms. It is provided solely for visual comparison purposes. The final position estimates of the MSCKF and the FMSCKF, along with the ground truth, are listed in Table 5.

#### 4.2.3. Long-range experiment

In the last experiment, we traversed a path spanning 2500 meters in a crowded neighborhood with a large number of moving cars and people. A sample image recorded during this experiment is shown in Figure 20. The ground-truth path along with the estimated paths using both the FMSCKF and the MSCKF algorithms is shown in Figure 21. Similar to the mid-range experiment, the ground-truth was obtained by reconstructing the trajectory of the vehicle based on the lanes it traveled

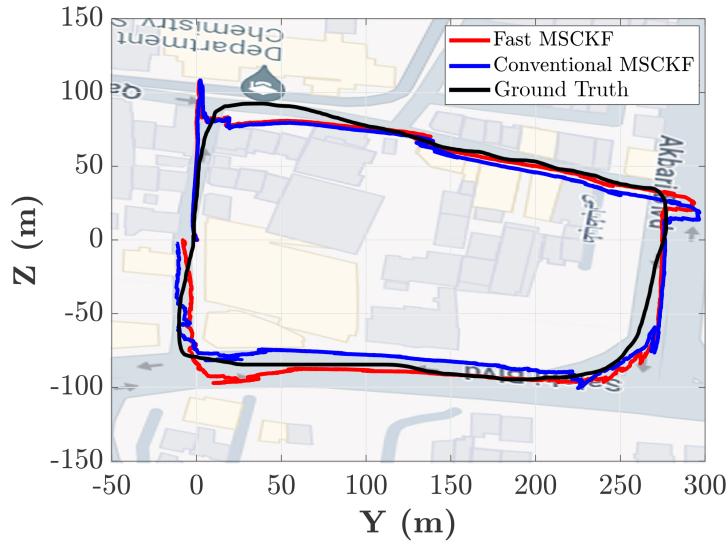


Figure 19: Estimated path in YZ plane using the MSCKF (blue) and the FMSCKF (red) along with the ground-truth path (black) in the mid-range experiment.

Table 5: Results of the mid-range experiment.

Algorithm	$x_f(m)$	$y_f(m)$	$z_f(m)$	Final point error (%)
FMSCKF	0	-3.3210	-0.7822	0.38
MSCKF	0	-4.512	-1.128	0.51
Ground truth	0	0	0	N.A.



Figure 20: A sample image of the long-range dataset.

in.

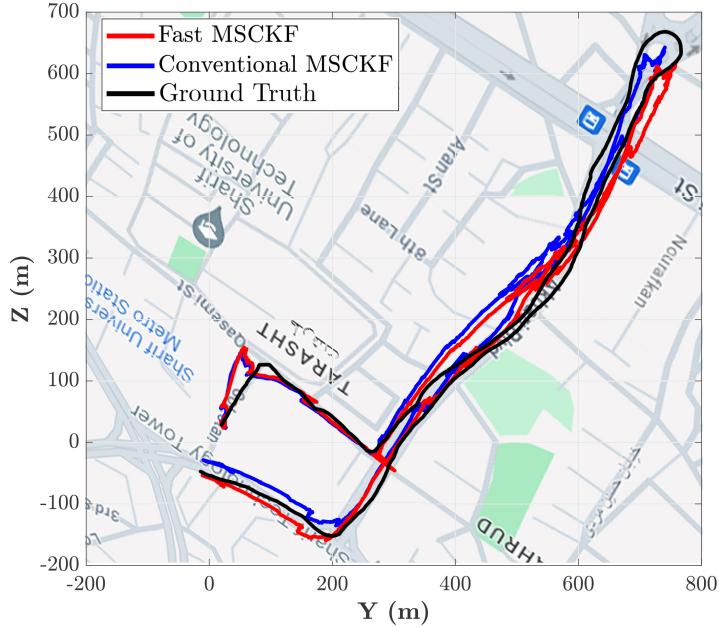


Figure 21: Estimated path in YZ plane using the FMSCKF (red) and the MSCKF (blue) along with the ground-truth path (black) in the long-range experiment.

The final position estimates of the MSCKF and the FMSCKF, as well as the ground truth, are listed in Table 6. Using the final point estimates in Table 6, it is straightforward to calculate the final point error for the algorithms. For the FMSCKF it is 0.41%, and for the MSCKF, it is 1.02% of the traveled distance. We observe that both algorithms perform well in long distances and in the presence of dynamic objects.

Table 6: Results of the long-range experiment.

Algorithm	$x_f(m)$	$y_f(m)$	$z_f(m)$	Final point error (%)
FMSCKF	0	-28.5850	-55.5651	0.41
MSCKF	0	-30.3751	-27.3251	0.51
Ground truth	0	-20	-50	N.A.

## 5. Discussion

The results in section 4 demonstrate the superiority of the FMSCKF over the original MSCKF in terms of both speed and accuracy. This superiority is explained in this section. The enhanced update rate in the FMSCKF can be attributed to two key factors. Firstly, the second scenario in the FMSCKF, where the number of features reaches its minimum, happens more frequently than the third one. Consequently, our method conducts more frequent pruning of the state vector and the covariance matrix compared to the original MSCKF, leading to a noticeable enhancement in algorithm speed. Secondly, in the conventional MSCKF, features are extracted from all frames, but in the FMSCKF, features are only extracted from the keyframes, when the number of tracked features is less than  $N_{f_{min}}$ . This strategic feature extraction methodology significantly reduces the computational cost associated with image processing in the FMSCKF.

As illustrated in Section 4, our approach exhibits comparable accuracy to the MSCKF in orientation estimation while surpassing it in position estimation. The reason for the higher accuracy in position in the FMSCKF is twofold. Firstly, the rate at which updates occur in the FMSCKF is higher compared to the original MSCKF. To elaborate, as mentioned in section 2.2, in the augmentation step, camera poses are calculated using the propagated IMU states. We know that the propagated states drift quickly, as IMU readings are integrated in that step. As a result, the longer the filter delays the update step, the less accurate camera poses will become, and consequently, the less accurate the triangulated points will be. Secondly, in the FMSCKF, when the number of tracked features falls below  $N_{f_{min}}$ , a relatively large number of features ( $N_{f_{min}}$ ) are used to do the update. Hence, the updated states are more accurate. It is worth noting that this particular event does not occur within the original MSCKF methodology.

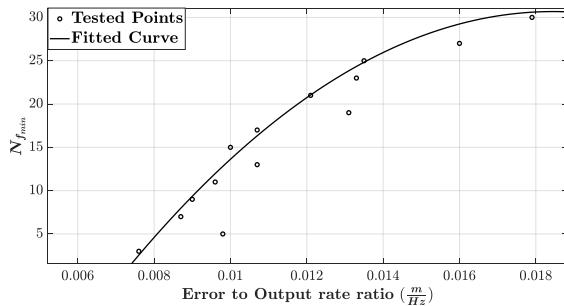


Figure 22: The effect of  $N_{f_{min}}$  on the error to speed ratio in the FMSCKF.

As discussed earlier,  $N_{f_{min}}$  is a tuning parameter and must be set appropriately. Setting  $N_{f_{min}}$  to high values causes high state pruning rates, thereby reducing the alge-

braic computational cost. However, a high value of  $N_{f_{min}}$  increases feature extraction frequency, which leads to a significantly higher image processing cost. Our extensive tests showed that the algebraic cost constitutes only 10% of the total computational cost, and the remaining 90% pertains to the image processing part. Moreover, a lower value of  $N_{f_{min}}$  corresponds to increased accuracy. This phenomenon occurs because when the minimum number of features is limited, the algorithm gains more information about individual features over consecutive image frames. In essence, the algorithm's knowledge about a feature increases as it is observed across a greater number of frames. Consequently, the triangulated point associated with the feature tends to be more precise.

As a result of the above discussion, it can be concluded that the lower  $N_{f_{min}}$  is set, the higher both the update rate and the accuracy will be. To validate the above-mentioned claim in practice, we analyzed the performance of the FMSCKF under different values of  $N_{f_{min}}$  using the first 10 seconds of the EuRoC MAV dataset MH\_01 (Burri et al., 2016). We calculated the ratio of *position error* to *output rate* for different values of  $N_{f_{min}}$ . A small ratio corresponds to either a small position error, a high output rate, or both, all of which are desirable. As a result, the introduced measure can be used to compare the performance of the proposed algorithm for different values of the parameter  $N_{f_{min}}$ . The smaller the ration, the better the performance. The result is shown in Figure 22. In accordance with our assertion, it is obvious from Figure 22 that when a higher  $N_{f_{min}}$  is used, the error-to-speed ratio is higher.

The reason for choosing the value 8 for  $N_{f_{min}}$  in our implementations is that in our algorithm, we use the well-known 8-point RANSAC algorithm for outlier rejection (Longuet-Higgins, 1981). To compute the fundamental matrix using this approach, we need at least 8 points. However, one can decide not to use this outlier rejection algorithm and choose a smaller value for  $N_{f_{min}}$ . According to Figure 22, this will result in an even higher accuracy and output rate.

## 6. Conclusion

In this paper, we aimed to address the challenge of fast and precise pose estimation (PE) for agile autonomous robots. We introduced an enhanced variant of the well-known Multi-State Constraint Kalman Filter (MSCKF), named Fast MSCKF (FMSCKF), designed to tackle the high computational cost associated with real-time implementation on resource-constrained robots. The FMSCKF leverages innovative feature marginalization and state pruning techniques to achieve computational efficiency, making it approximately six times faster than the standard MSCKF, all while delivering a substantial improvement of at least 20% in final position estimation accuracy. In section 5, we thoroughly discussed the differences between the

MSCKF and the FMSCKF and highlighted the reasons why the FMSCKF outperforms the MSCKF. Our extensive evaluation of the FMSCKF on both public datasets and in real-world experiments demonstrated the remarkable performance of the FMSCKF compared to the state-of-the-art algorithms. Future works include exploring machine-learning-based feature extraction and tracking algorithms and evaluating their performance and robustness in VIO applications. Another direction could be using additional sensors, such as LiDAR, to enhance the performance of the proposed algorithm.

## References

- Abbasi, E., Ghayour, M., Danesh, M., Amiri, P., Yoosefian, M.H., 2018a. Formation flight control and path tracking of a multi-quadrotor system in the presence of measurement noise and disturbances, in: 2018 6th RSI International Conference on Robotics and Mechatronics (IcRoM), IEEE. pp. 273–279.
- Abbasi, E., Ghayour, M., Danesh, M., Yoosefian, M.H., 2018b. Optimal path tracking of a quadrotor in the presence of obstacle using the league championship algorithm, in: 2018 6th RSI International Conference on Robotics and Mechatronics (IcRoM), IEEE. pp. 236–242.
- Abdollahi, M., Banazadeh, A., Pourtakdoust, S., 2019. Experimental investigation of vertical cg position changes on quadrotor's performance via frequency-domain identification techniques. Sharif Journal of Mechanical Engineering 35, 129–138.
- Azimi, A., Ahmadabadian, A.H., Remondino, F., 2022. Pks: A photogrammetric key-frame selection method for visual-inertial systems built on orb-slam3. ISPRS Journal of Photogrammetry and Remote Sensing 191, 18–32.
- Bloesch, M., Burri, M., Omari, S., Hutter, M., Siegwart, R., 2017. Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback. The International Journal of Robotics Research 36, 1053–1072.
- Brossard, M., Bonnabel, S., Barrau, A., 2018. Invariant kalman filtering for visual inertial slam, in: 2018 21st International Conference on Information Fusion (FUSION), IEEE. pp. 2021–2028.
- Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M.W., Siegwart, R., 2016. The euroc micro aerial vehicle datasets. The International Journal of Robotics Research 35, 1157–1163.

- Cen, R., Zhang, X., Tao, Y., Xue, F., Zhang, Y., 2020. Temporal delay estimation of sparse direct visual inertial odometry for mobile robots. *Journal of the Franklin Institute* 357, 3893–3906.
- Chen, C., Wang, B., Lu, C.X., Trigoni, N., Markham, A., 2020. A survey on deep learning for localization and mapping: Towards the age of spatial machine intelligence. *arXiv preprint arXiv:2006.12567* .
- Cheng, C., Li, X., Xie, L., Li, L., 2022. Autonomous dynamic docking of uav based on uwb-vision in gps-denied environment. *Journal of the Franklin Institute* 359, 2788–2809.
- Delmerico, J., Scaramuzza, D., 2018. A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots, in: 2018 IEEE international conference on robotics and automation (ICRA), IEEE. pp. 2502–2509.
- Engel, J., Koltun, V., Cremers, D., 2017. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence* 40, 611–625.
- Fischler, M.A., Bolles, R.C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* 24, 381–395.
- Forster, C., Carbone, L., Dellaert, F., Scaramuzza, D., 2016a. On-manifold preintegration for real-time visual-inertial odometry. *IEEE Transactions on Robotics* 33, 1–21.
- Forster, C., Zhang, Z., Gassner, M., Werlberger, M., Scaramuzza, D., 2016b. Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics* 33, 249–265.
- Heo, S., Cha, J., Park, C.G., 2018. Ekf-based visual inertial navigation using sliding window nonlinear optimization. *IEEE Transactions on Intelligent Transportation Systems* 20, 2470–2479.
- Hesch, J.A., Kottas, D.G., Bowman, S.L., Roumeliotis, S.I., 2013. Consistency analysis and improvement of vision-aided inertial navigation. *IEEE Transactions on Robotics* 30, 158–176.
- Huang, G., 2019. Visual-inertial navigation: A concise review, in: 2019 international conference on robotics and automation (ICRA), IEEE. pp. 9572–9582.

- Huang, G.P., Mourikis, A.I., Roumeliotis, S.I., 2009. A first-estimates jacobian ekf for improving slam consistency, in: Experimental Robotics: The Eleventh International Symposium, Springer. pp. 373–382.
- Li, C., Waslander, S.L., 2020. Towards end-to-end learning of visual inertial odometry with an ekf, in: 2020 17th Conference on Computer and Robot Vision (CRV), IEEE. pp. 190–197.
- Li, M., Mourikis, A.I., 2013. High-precision, consistent ekf-based visual-inertial odometry. *The International Journal of Robotics Research* 32, 690–711.
- Longuet-Higgins, H.C., 1981. A computer algorithm for reconstructing a scene from two projections. *Nature* 293, 133–135.
- Loo, S.Y., Amiri, A.J., Mashohor, S., Tang, S.H., Zhang, H., 2019. Cnn-svo: Improving the mapping in semi-direct visual odometry using single-image depth prediction, in: 2019 International conference on robotics and automation (ICRA), IEEE. pp. 5218–5223.
- Ma, F., Shi, J., Yang, Y., Li, J., Dai, K., 2019. Ack-msckf: Tightly-coupled ackermann multi-state constraint kalman filter for autonomous vehicle localization. *Sensors* 19, 4816.
- Macias, V., Becerra, I., Martinez, E., Murrieta-Cid, R., Becerrra, H.M., 2021. Single landmark feedback-based time optimal navigation for a differential drive robot. *Journal of the Franklin Institute* 358, 4761–4792.
- Mourikis, A.I., Roumeliotis, S.I., 2007. A multi-state constraint kalman filter for vision-aided inertial navigation, in: Proceedings 2007 IEEE international conference on robotics and automation, IEEE. pp. 3565–3572.
- Mur-Artal, R., Montiel, J.M.M., Tardos, J.D., 2015. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics* 31, 1147–1163.
- Nousias, S., Lourakis, M., Bergeles, C., 2019. Large-scale, metric structure from motion for unordered light fields, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3292–3301.
- Qin, T., Li, P., Shen, S., 2018. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics* 34, 1004–1020.
- Rehder, J., Nikolic, J., Schneider, T., Hinzmann, T., Siegwart, R., 2016. Extending kalibr: Calibrating the extrinsics of multiple imus and of individual axes, in: 2016

IEEE International Conference on Robotics and Automation (ICRA), IEEE. pp. 4304–4311.

Rosinol, A., Abate, M., Chang, Y., Carlone, L., 2020. Kimera: an open-source library for real-time metric-semantic localization and mapping, in: 2020 IEEE International Conference on Robotics and Automation (ICRA), IEEE. pp. 1689–1696.

Sola, J., 2017. Quaternion kinematics for the error-state kalman filter. arXiv preprint arXiv:1711.02508 .

Sukkarieh, S., Nebot, E.M., Durrant-Whyte, H.F., 1999. A high integrity imu/gps navigation loop for autonomous land vehicle applications. IEEE transactions on robotics and automation 15, 572–578.

Sun, K., Mohta, K., Pfrommer, B., Watterson, M., Liu, S., Mulgaonkar, Y., Taylor, C.J., Kumar, V., 2018. Robust stereo visual inertial odometry for fast autonomous flight. IEEE Robotics and Automation Letters 3, 965–972.

Tomasi, C., Kanade, T., 1993. Shape and motion from image streams: a factorization method. Proceedings of the National Academy of Sciences 90, 9795–9802.

Triggs, B., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.W., 2000. Bundle adjustment—a modern synthesis, in: Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms Corfu, Greece, September 21–22, 1999 Proceedings, Springer. pp. 298–372.

Zheng, X., Moratto, Z., Li, M., Mourikis, A.I., 2017. Photometric patch-based visual-inertial odometry, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE. pp. 3264–3271.

Zuo, X., Merrill, N., Li, W., Liu, Y., Pollefeys, M., Huang, G., 2021. Codevio: Visual-inertial odometry with learned optimizable dense depth, in: 2021 IEEE International Conference on Robotics and Automation (ICRA), IEEE. pp. 14382–14388.