# Efficient Model-Aided Visual-Inertial Ego-Motion Estimation for Multirotor MAVs

Xu, Y.; de Croon, G.C.H.E.

**Publication date**
2023

**Document Version**
Final published version

**Published in**
14th annual international micro air vehicle conference and competition

# Efficient Model-Aided Visual-Inertial Ego-Motion Estimation for Multirotor MAVs

Yingfu Xu and Guido C. H. E. de Croon *
Delft University of Technology



ABSTRACT

When deployed onboard micro air vehicles (MAVs) with limited processing power, visual ego-motion estimation solutions face an efficiency-accuracy trade-off. This paper proposes an aerodynamic-model-aided approach that emphasizes time efficiency over estimation accuracy. A linear drag force model of propellers guarantees bounded estimation errors in the velocity components orthogonal to the shafts of propellers and the attitude relative to the gravity direction. Feature point correspondences are extracted from the monocular image stream to compute the relative heading angle and translational direction, which is fused with inertial measurements by an extended Kalman filter (EKF) in a loosely coupled manner. The proposed approach shows balanced performance in accuracy and efficiency. It also has robustness to situations where vision information becomes unavailable.

## 1 INTRODUCTION

The autonomous flight of micro air vehicles (MAVs) in GPS-denied environments is challenging. For large flying robots (*i.e.* heavier than ∼300 grams, and larger than ∼50 centimeters in diameter), this task is substantially simpler since they are less constraint by computational or payload capacity, and can process information from numerous sensors. However, they require more flying space and stringent safety checks, and are generally more expensive. In contrast, palm-sized or smaller MAVs [1] are promising alternatives as being safer and cheaper. But they come with the disadvantage of limited sensing and processing resources. It makes monocular visual-inertial odometry (VIO) that combines a single camera and an inertial measurement unit (IMU) the most promising option for ego-motion estimation. The camera provides abundant up-to-scale information about the surroundings, and the IMU measures metric-scale rotational rate and translational acceleration in high frequency.

VIO [2, 3, 4, 5] has shown good performance in MAV navigation in recent years. Ego-motion estimation and environment mapping can be performed simultaneously by taking the reprojection error from observations of the same landmark in different frames and the integrated IMU measurements as constraints between the camera poses and landmarks' locations. The locations of landmarks are treated as states to be estimated, along with camera poses. Environment mapping not only contributes to the accuracy of the ego-motion estimation, but also provides an obstacle map for motion planning. However, this comes at the cost of iterative computation and the need for an initialization procedure, which may be required multiple times in case of tracking failure. Even for approaches [3] that map sparse-point feature points and use a sliding window to bound the optimization, the computational demand of this task is still considerable.

Focusing on ego-motion, the multi-state constraint Kalman filter (MSCKF) [2] maintains a window of camera poses but does not optimize landmark location. It is based on an extended Kalman filter (EKF) whose state vector is augmented with poses of the previous frames that observed the same feature points. The least-squares location of a point is only calculated once after it is no longer visually tracked. Its reprojection errors, which express geometric constraints between the camera poses, are then used in visual measurement updates. Instead of reprojecting 3D point locations into multiple frames, [5] enforces multiple constraints on the relative pose of a frame pair through the two observations of the same point. The drift in the relative pose is reflected by the residual of the epipolar geometry constraint. This residual updates the paired camera poses in an EKF. The absence of point locations reduces computational demand, and only one frame's pose is needed to augment the state vector. However, point correspondences perform updating one by one using the raw pixel locations of tracked points. The estimation accuracy can suffer from frame-to-frame feature tracking noise.

The above-mentioned approaches use the pixel locations of feature points. Visual and inertial measurements are fused in a tightly-coupled manner. Visual odometry (VO) systems like the SVO [6] produce up-to-scale environment map and ego-motion estimation that can be loosely coupled with IMU measurements by an EKF [7]. The EKF-based, loosely-coupled approach SVO+MSF [8] and the EKF-based, tightly-coupled, map-less MSCKF outperform others in efficiency among several VIO solutions [9].

In windless environments, since all the aerodynamic force acting on an MAV is caused by propeller rotation and ego-motion, the aerodynamic model can be an information source for ego-motion estimation. As shown in [10], a simplified linear drag model is combined with IMU measurements to es-

*Email addresses: yingfu.xu.94@gmail.com, g.c.h.e.decroon@tudelft.nl

timate horizontal components of attitude and velocity in the multirotor's body frame. Unlike estimation from purely integrated IMU measurements, the error of this model-aided estimation does not increase over time and is related to IMU bias and model fidelity. This aerodynamic model was used for velocity prediction [1, 5] and trajectory tracking [11]. In [12], high estimation accuracy was reached with a more precise dynamic model that takes into account thrust forces and the effect of rotor speed on the drag, which makes the vertical speed in the body frame to be observable. The camera provides point correspondences only to estimate the relative yaw angle between a pair of frames. Worth noticing is that the drift-free roll and pitch angles were used as known values to simplify the calculation of the relative pose.

To gain robustness and accuracy with as little computation as possible, we propose an approach that combines all the previously mentioned strategies that benefit time efficiency. It is a map-less, model-aided, EKF-based, loosely-coupled ego-motion estimator for multirotor MAVs. The linear drag force model prevents attitude and velocity estimation in the body's horizontal plane from drifting over time when there is no visual information. The relative heading angle and the direction of translational motion between two frames are calculated from visual feature point correspondences using the epipolar constraint. The attitude estimation is taken as known information to simplify the visual pose calculation. The visual update executes in a one-frame-one-time manner.

The proposed ego-motion estimator has a relatively low-complexity modular pipeline that is easy to implement and debug. We choose the EuRoC [13] dataset as the validation tool and compare our approach with MSCKF [2], a relatively efficient (yet accurate) VIO solution. The accuracy of the proposed approach is compromised due to the prioritized efficiency. But it is sufficient for short-time navigation. And it is observed that the estimator can maintain its accuracy when visual information is no longer available.

## 2 ESTIMATOR FRAMEWORK

The proposed ego-motion estimator pipeline is shown in Fig. 1. The *Visual Processing* module receives estimated roll and pitch angles from the *EKF* module. The relative pose is calculated from the feature point correspondences in the current frame and the keyframe based on the estimated roll and pitch angles. The relative yaw angle and the direction of translation with respect to the keyframe are taken as measurements for the visual update step of the EKF.

### 2.1 Definitions

In this paper, we denote all scalars by lowercase letters $x$, vectors by lowercase bold letters $\boldsymbol{x}$, and matrices by bold uppercase letters $\boldsymbol{X}$. Coordinate frames are denoted by non-bold uppercase letters $X$. Estimated values are written as $\hat{x}$, and raw measurements as $\tilde{x}$.

As shown in Fig. 2, the body frame $B$ (green) is defined according to the pose of the propellers. The plane defined

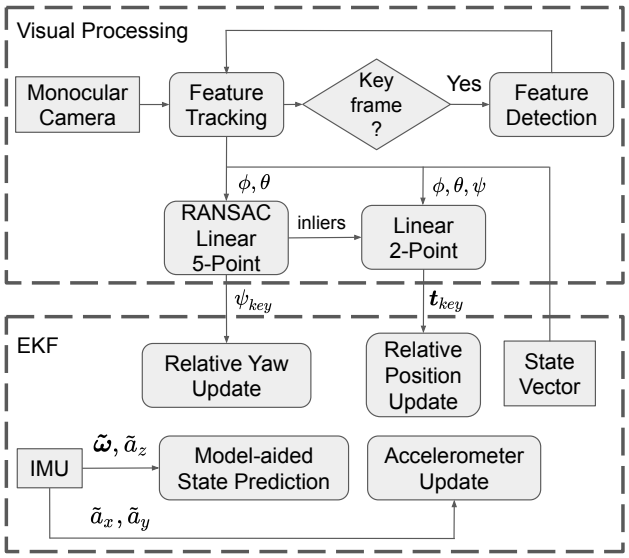



Figure 1: Pipeline of the proposed approach.

by the $x$-axis and $y$-axis of $B$ is orthogonal to the propellers' rotation axis. The IMU is located at the origin of $B$. IMU measurements are required to be expressed in $B$ before being used by the estimator. The world frame $W$ (black) is stationary. Its $z$-axis $z_W$ has the same direction as gravity. The heading frame $H$ (red) is defined by rotating the world frame around the $z$-axis by the yaw angle. Thus $z_H$ always points to the direction of gravity. The heading frame's origin coincides with the origin of the camera frame $C$. The Euler angles roll $\phi$, pitch $\theta$, and yaw $\psi$ reflect the rotation between $W$ and $B$. The rotation sequence is first $z$-axis ($\psi$), then $y$-axis ($\theta$), and last $x$-axis ($\phi$). $\phi$ and $\theta$ reflect the attitude of $B$ respect to the gravity diretion.

If the number of tracked feature points is below the preset threshold, or there are not enough inlier points in the RANSAC-based calculation of the essential matrix, the current frame is defined as the new keyframe. A certain number of feature points are detected in a new keyframe. Unit vector $\boldsymbol{t}_{key}$ points from the camera position of the keyframe (right) to the current camera position (left), as shown in Fig. 2. The solid lines connect the camera with the feature points detected in the keyframe. The dashed lines connect the camera with points tracked in the current frame.

For cameras mounted far from the body center, we consider the translational velocity of the camera caused by its rotation relative to IMU, which is neglected in [5, 12]. The vector that points from the IMU to the camera expressed in the body frame is denoted by $\boldsymbol{p}_B^C$.

### 2.2 Linear Drag Model

There are various aerodynamic forces acting on an multirotor MAV in flight. In [12], the thrust produced is modelled to be proportional to the sum of the squares of pro-

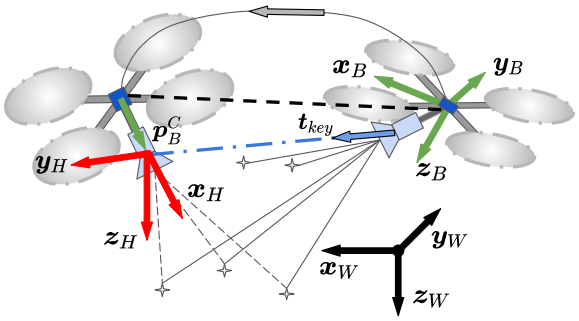



Figure 2: Schematic that illustrates the definition of the different coordinate frames and the motion of the MAV from the key frame pose to the current pose.

peller speed. The drag acting on propellers is proportional to the product of the MAV's velocity and the sum of the rotor speeds. The velocity-square-proportional parasitic drag on the airframe was considered in [14] to improve the accuracy of the model in high-speed flight.

With the sensor setup that only consists of an IMU and a camera, we choose the simplified velocity-proportional drag force model [10]. We define the propeller plane as the plane orthogonal to the shafts of propellers. The drag force vector inside the propeller plane is approximately proportional to the projection of the MAV's velocity vector to the propeller plane. The drag parameter $k_d$ is estimated as an EKF state to compensate for its variation in different aerodynamic regimes. Its derivative is modeled as a small Gaussian white noise $\boldsymbol{w}_{k_d}$.

The biased and noisy IMU measurements are modeled as

$$\tilde{\boldsymbol{a}} = \hat{\boldsymbol{a}} + \boldsymbol{b}_a + \boldsymbol{w}_a, \quad \tilde{\boldsymbol{\omega}} = \hat{\boldsymbol{\omega}} + \boldsymbol{b}_g + \boldsymbol{w}_g, \tag{1}$$

where $\boldsymbol{w}_a$ and $\boldsymbol{w}_g$ are white Gaussian noise. We model the derivatives of the additive accelerometer bias $\boldsymbol{b}_a$ and gyroscope bias $\boldsymbol{b}_g$ as small white Gaussian noise $\boldsymbol{w}_{\boldsymbol{b}_a}$ and $\boldsymbol{w}_{\boldsymbol{b}_g}$, respectively. $\hat{\boldsymbol{a}}$ denotes the translational acceleration caused by the aerodynamic force acting on a flying MAV, mainly consisting of the drag force and the thrust acting on the propellers. Hence

$$\begin{aligned}\hat{\boldsymbol{a}} &= [a_x, a_y, a_z]^T \\ &= [k_d v_{B,x} + w_{v,x}, \quad k_d v_{B,y} + w_{v,y}, \\ &\quad \tilde{a}_z - b_{a,z} + w_{a,z}]^T\end{aligned} \tag{2}$$

where $\boldsymbol{w}_{\boldsymbol{v}}$ denotes the noise of the linear drag model.

### 2.3 State Propagation

The estimator back-end is a simplified variant of the EKF-based back-end of the robocentric VIO [15]. It estimates the relative pose between the current body frame and the local frame of reference. The EKF state vector is defined as

$$\begin{aligned}\boldsymbol{x} := [&{}^{R_k}\boldsymbol{p}_G, \quad {}^{R_k}_{G}\boldsymbol{q}, \quad \boldsymbol{g}_{R_k}, \\ &{}^{B_t}\boldsymbol{p}_{R_k}, \quad {}^{B_t}_{R_k}\boldsymbol{q}, \quad \boldsymbol{v}_{B_t}, \quad \boldsymbol{b}_a, \quad \boldsymbol{b}_g, \quad k_d].\end{aligned} \tag{3}$$

$B_t$ is the current body frame at time $t$. When a new keyframe is defined, the new reference frame $R_{k+1}$ is set to be the same as the $B_t$ at the time. $R_k$ is the current reference frame. It is the $k$th reference frame since the estimator is initialized. $G$ stands for the global frame. It is the first reference frame $R_0$, *i.e.*, the body frame when the first keyframe is captured after the initialization of the estimator. Note that $G$ and $W$ are not the same coordinate frame. They have the same origin point, but there is relative rotation between them. $\boldsymbol{R}({}^{G}_{W}\boldsymbol{q})$ can be calculated from $\boldsymbol{R}({}^{G}_{W}\boldsymbol{q}) \cdot \boldsymbol{R}({}^{R_k}_{G}\boldsymbol{q}) \cdot \boldsymbol{g}_{R_k} = [0, 0, g]^T$. ${}^{R_k}\boldsymbol{p}_G$ is a translation vector pointing from the origin of $G$ to the origin of $R_k$, expressed in $R_k$. It is about the global position of $R_k$. ${}^{B_t}\boldsymbol{p}_{R_k}$ is a translation vector pointing from the origin of $R_k$ to the origin of $B_t$, expressed in $B_t$. It is about the local position of $B_t$ relative to $R_k$. ${}^{R_k}_{G}\boldsymbol{q}$ is the Hamilton quaternion reflecting the relative rotation between $G$ and $R_k$. ${}^{B_t}_{R_k}\boldsymbol{q}$ reflects the relative rotation between $R_k$ and $B_t$. $\boldsymbol{g}_{R_k}$ indicates the gravity vector expressed in $R_k$. $\boldsymbol{v}_{B_t}$ is the translational velocity of the IMU expressed in $B_t$.

Eq. (4) shows the IMU-driven state dynamics ($\dot{\boldsymbol{x}}$). $[\hat{\boldsymbol{\omega}}]_\times$ represents the skew-symmetric matrix associated with $\hat{\boldsymbol{\omega}}$. $\boldsymbol{w}_{\boldsymbol{p}}$ is the process noise in position integration. $\boldsymbol{R}({}^{B_t}_{R_k}\boldsymbol{q})$ is a transformation function from ${}^{B_t}_{R_k}\boldsymbol{q}$ to *SO3* rotation matrix that maps a vector expressed in $B_t$ to its expression in $R_k$. $\otimes$ denotes quaternion product. $\boldsymbol{E}_z$ is a $3 \times 3$ diagonal matrix with $[0, 0, 1]$ as its diagonal elements. $\boldsymbol{E}_{x,y}$ is a $3 \times 3$ diagonal matrix with $[1, 1, 0]$ as its diagonal elements. We utilize the techniques introduced in [16] for quaternion-related calculation.

$$\begin{aligned}{}^{B_t}\dot{\boldsymbol{p}}_{R_k} &= -[\hat{\boldsymbol{\omega}}]_\times \cdot {}^{B_t}\boldsymbol{p}_{R_k} + \boldsymbol{v}_{B_t} + \boldsymbol{w}_{\boldsymbol{p}}, \\ \dot{\boldsymbol{v}}_{B_t} &= -[\hat{\boldsymbol{\omega}}]_\times \cdot \boldsymbol{v}_{B_t} + \boldsymbol{R}({}^{B_t}_{R_k}\boldsymbol{q})^T \cdot \boldsymbol{g}_{R_k} \\ &\quad + \boldsymbol{E}_{x,y}(k_d \boldsymbol{v}_{B_t} + \boldsymbol{w}_{\boldsymbol{v}}) + \boldsymbol{E}_z \hat{\boldsymbol{a}}, \\ {}^{B_t}_{R_k}\dot{\boldsymbol{q}} &= \frac{1}{2} {}^{B_t}_{R_k}\boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \hat{\boldsymbol{\omega}} \end{bmatrix}, \\ \dot{\boldsymbol{b}}_a &= \boldsymbol{w}_{\boldsymbol{b}_a}, \quad \dot{\boldsymbol{b}}_g = \boldsymbol{w}_{\boldsymbol{b}_g}.\end{aligned} \tag{4}$$

### 2.4 Acceleration Measurement Update

Drag-induced acceleration in the horizontal body plane (the plane that contains the $x$-axis and $y$-axis of $B$) can be measured by accelerometer measurements along the $x$-axis and $y$-axis of $B$. Accelerometer measurements correct state propagation through the following measurement update equations of the EKF.

$$\begin{aligned}z_{a,x} &= k_d v_{B,x} + b_{a,x} + w_{a,x}, \\ z_{a,y} &= k_d v_{B,y} + b_{a,y} + w_{a,y}.\end{aligned} \tag{5}$$

The accelerometer measurement update steps along with the model-aided inertial ego-motion state propagation. The estimated horizontal states ($\boldsymbol{g}_{R_k}$, $v_{B,x}$, and $v_{B,y}$) do not drift over time. But they are noisy and negatively affected by the accelerometer bias $\boldsymbol{b}_a$. The performance of model-aided inertial ego-motion estimation is omitted in this paper. An interested reader can refer to [10]. We use the drift-free roll $\hat{\phi}$ and

pitch $\hat{\theta}$ angles to simplify the visual measurement processing, as introduced in Section 3. $\hat{\phi}$ and $\hat{\theta}$ of the current frame are calculated from $\boldsymbol{R}({}^{B_t}_{R_k}\boldsymbol{q})^T \cdot \boldsymbol{g}_{R_k} = R(\hat{\phi}) \cdot R(\hat{\theta}) \cdot [0, 0, g]^T$.

### 2.5 Relative Visual Measurement Update

The *Visual Processing* module outputs the visual measurements of the relative pose with respect to the keyframe, $\hat{\psi}_{key}$ and ${}^{H,key}\hat{\boldsymbol{t}}$, to be introduced in Section 3. $\hat{\psi}_{key}$ reflects the visual measurement of the 1-dimension rotation between the current heading frame and the keyframe heading frame. ${}^{H,key}\hat{\boldsymbol{t}}$ is the visual measurement of the camera translation direction expressed in the keyframe heading frame. The EKF uses these measurements in the visual measurement update through Eq. (6).

$$\begin{aligned} \boldsymbol{z_r} &= {}^{B_t}_{R_k}\boldsymbol{q} + \boldsymbol{w_r}, \\ \boldsymbol{z_t} &= \frac{\boldsymbol{t}_{BC} + {}^{B_t}\boldsymbol{p}_{R_k} - \boldsymbol{R}({}^{B_t}_{R_k}\boldsymbol{q})^T \cdot \boldsymbol{t}_{BC}}{\|\boldsymbol{t}_{BC} + {}^{B_t}\boldsymbol{p}_{R_k} - \boldsymbol{R}({}^{B_t}_{R_k}\boldsymbol{q})^T \cdot \boldsymbol{t}_{BC}\|} + \boldsymbol{w_t}. \end{aligned} \tag{6}$$

$\boldsymbol{z_r}$ is the rotation between the current body frame and the keyframe body frame. It is calculated from $\hat{\psi}_{key}$ together with the roll and pitch angles of the current frame $\hat{\phi}_{curr.}, \hat{\theta}_{curr.}$ and the keyframe $\hat{\phi}_{key}, \hat{\theta}_{key}$. $\boldsymbol{z_t}$ is the direction of the translational motion from the camera position at the keyframe to the current camera position, expressed in the current body frame. It is obtained by rotating ${}^{H,key}\hat{\boldsymbol{t}}$ to get its expression in the current body frame via $\hat{\phi}_{curr.}$, $\hat{\theta}_{curr.}$, and $\hat{\psi}_{key,post.}$. $\boldsymbol{z_t}$ is a unit vector, so the propagated camera translation is normalized accordingly. $\boldsymbol{t}_{BC}$ is the translation vector points from the IMU to the camera, expressed in $B$. In the implementation, we neglect the correlation of the elements in the three axes of the visual measurements. Thus the measurement noise matrices $\boldsymbol{R}_{\boldsymbol{w_t}}$ and $\boldsymbol{R}_{\boldsymbol{w_r}}$ are diagonal matrices.

The rotation update using $\boldsymbol{z_r}$ takes place first. And then, translation update uses $\boldsymbol{z_t}$. The purpose of this two-step updating is to benefit the calculation of the visual translation measurement ${}^{H,key}\hat{\boldsymbol{t}}$ by the more accurate *a posteriori* relative yaw angle $\hat{\psi}_{key,post.}$, as introduced in Section 3. $\hat{\psi}_{key,post.}$ is calculated from the *a posteriori* ${}^{R_k}_{G}\hat{\boldsymbol{q}}$ and $\hat{\boldsymbol{g}}_{R_k}$ that has been updated by $\boldsymbol{z_r}$.

### 2.6 Composition and Resetting for New Keyframe

When a new keyframe is defined, the *a posterior* relative pose estimation ${}^{B_t}\hat{\boldsymbol{p}}_{R_k}$ and ${}^{B_t}_{R_k}\hat{\boldsymbol{q}}$ are composed to the global pose, as shown in Eq. (7).

$$\begin{aligned} {}^{R_{k+1}}_{G}\boldsymbol{q} &= {}^{B_t}_{R_k}\hat{\boldsymbol{q}} \otimes {}^{R_k}_{G}\boldsymbol{q}, \\ {}^{R_{k+1}}\boldsymbol{p}_G &= \boldsymbol{R}({}^{B_t}_{R_k}\hat{\boldsymbol{q}})^T \cdot {}^{R_k}\boldsymbol{p}_G + {}^{B_t}\hat{\boldsymbol{p}}_{R_k} \end{aligned} \tag{7}$$

The current body frame $B_t$ becomes the new reference frame $R_{k+1}$. The expression of the gravity vector $\boldsymbol{g}_{R_{k+1}}$ in $R_{k+1}$ is calculated as $\boldsymbol{g}_{R_{k+1}} = \boldsymbol{R}({}^{B_t}_{R_k}\hat{\boldsymbol{q}})^T \cdot \hat{\boldsymbol{g}}_{R_k}$. ${}^{B_t}\boldsymbol{p}_{R_{k+1}}$ and ${}^{B_t}_{R_{k+1}}\boldsymbol{q}$ are set to a zero vector and a unit quaternion whose vector part is a zero vector, respectively. Their corresponding elements in the covariance matrix are set to zeros too.

## 3 VISUAL RELATIVE POSE ESTIMATION

In this section, we describe the proposed method for estimating the yaw angle and the direction of translation relative to the keyframe. The relative yaw and translation are calculated separately, as shown in Fig. 1.

### 3.1 Keyframe-based Feature Tracking

We follow the same idea of keyframe-based feature detection and tracking strategy as [5, 12]. Kanade-Lucas-Tomasi (KLT) [17] is utilized as the feature tracker to continuously track FAST features [18] between frames. If the number of points tracked in the current frame falls below a threshold, the current frame is defined as the new keyframe. A fixed number of uniformly distributed FAST points are detected in a new keyframe by evenly splitting the image into several regions while keeping the same number of good features in each region. These newly detected points are then added to a database for tracking in the coming frames.

### 3.2 Linear Relative Yaw Calculation

Based on the epipolar geometry of a pair of corresponding points, the linear 8-point algorithm [19] calculates the essential matrix which helps extracting the 3D relative rotation and the 2D direction of translation. If two relative orientation angles are known, the remaining angle and the direction of translation can be calculated linearly with 5-point pairs [20]. Slightly different from [12], our linear 5-point algorithm projects point coordinates on a unit sphere in the heading frame $H$.

The pixel location of a feature point is first undistorted and normalized using the distortion parameters and the intrinsic matrix of the camera. We then obtain the homogeneous coordinates of the feature points expressed in the camera frame, ${}^{C}\tilde{\boldsymbol{p}}_{key}$ and ${}^{C}\tilde{\boldsymbol{p}}_{curr.}$. The epipolar constraint is given by

$${}^{C}\tilde{\boldsymbol{p}}^T_{curr.} \lfloor {}^{C,curr.}\boldsymbol{t}_{\times} \rfloor \boldsymbol{R}^{C,curr.}_{C,key} {}^{C}\tilde{\boldsymbol{p}}_{key} = 0 \tag{8}$$

where $\lfloor {}^{C,curr.}\boldsymbol{t}_{\times} \rfloor$ is the skew-symmetric matrix of the translation vector ${}^{C,curr.}\boldsymbol{t}_{key}$ that points from the origin of the current camera frame to the origin of the keyframe camera frame, expressed in current camera frame.

Using the camera extrinsic rotation matrix $\boldsymbol{R}^B_C$ and the estimated roll $\hat{\phi}$ and pitch $\hat{\theta}$ angles of the keyframe and the current frame, we can express the feature point coordinates in the heading frame $H$, as

$$\begin{aligned} {}^{H}\hat{\boldsymbol{p}}_{key} &= \boldsymbol{R}^T_{\hat{\theta},key} \boldsymbol{R}^T_{\hat{\phi},key} \boldsymbol{R}^B_C {}^{C}\tilde{\boldsymbol{p}}_{key}, \\ {}^{H}\hat{\boldsymbol{p}}_{curr.} &= \boldsymbol{R}^T_{\hat{\theta},curr.} \boldsymbol{R}^T_{\hat{\phi},curr.} \boldsymbol{R}^B_C {}^{C}\tilde{\boldsymbol{p}}_{curr.}. \end{aligned} \tag{9}$$

${}^{H}\hat{\boldsymbol{p}}_{key}$ and ${}^{H}\hat{\boldsymbol{p}}_{curr.}$ are normalized to unit vectors ${}^{H}\overline{\boldsymbol{p}}_{key}$ and ${}^{H}\overline{\boldsymbol{p}}_{curr.}$, which are equivalent to the 3D coordinates of the feature points projected onto the unit sphere. They describe the directions of the feature points in the heading frame $H$,

so we refer to them as *point vectors*. The epipolar constraint equation in $H$ is given by

$$ {}^{H}\overline{\boldsymbol{p}}_{curr.}^{T} \lfloor {}^{H,curr.}\boldsymbol{t}_{\times} \rfloor \boldsymbol{R}_{H,key}^{H,curr.} {}^{H}\overline{\boldsymbol{p}}_{key} = 0 \qquad (10) $$

where $\boldsymbol{R}_{H,key}^{H,curr.}$ is the relative rotation between two heading frames, which equals to $\boldsymbol{R}_{\psi_{key}}$. In Eq. (10), the essential matrix $\boldsymbol{E}_H = \lfloor {}^{H,curr.}\boldsymbol{t}_{\times} \rfloor \boldsymbol{R}_{\psi_{key}}$ has six entries defined up-to-scale, which can be linearly solved with a minimum of five points [20]. The visual measurement of relative yaw $\hat{\psi}_{key}$ is then calculated from $\boldsymbol{E}_H$.

Both [5] and [12] reject outliers in the tracked points using the prior relative pose based on state propagation. They apply a threshold on the 2D distance from each tracked point to the epipolar line or the residuals of the epipolar constraint equation. This scheme is faster than iterative random sample consensus (RANSAC) outlier rejection. However, aggressive maneuvers may cause fewer feature points to be detected and correctly tracked. Due to the biases of the IMU measurements and the simplified drag model, the propagated relative translational motion would diverge if without enough vision updates. Consequently, the prior relative pose is not accurate enough for outlier rejection. Therefore, we employ RANSAC in the linear 5-point algorithm to calculate the relative yaw angle. Setting the number of random trials to a reasonably small number limits the required computation cost.

### 3.3 Linear Relative Translation Direction Calculation

During implementing the 5-point algorithm and testing it on EuRoC, it was observed that the translation direction vectors calculated together with relative yaw were noisy. The standard deviation of the direction error can be more than 30 degrees compared to the ground-truth motion direction. Instead of using this translational direction in the EKF update, we take the relative yaw angle as a known value and re-calculate the translation vector ${}^{H,key}\hat{\boldsymbol{t}}$. As introduced before, the visual translation measurement update happens after the rotation update. So we can calculate ${}^{H,key}\hat{\boldsymbol{t}}$ after the rotation update and make use of the updated *a posteriori* relative yaw angle $\hat{\psi}_{key,post.}$. This is achieved by rotating ${}^{H}\overline{\boldsymbol{p}}_{curr.}$ from the current heading frame to keyframe heading frame by $\boldsymbol{R}(\hat{\psi}_{key,post.})$. Then we get the epipolar constraint equation in the keyframe heading frame as

$$ {}^{H,key}\overline{\boldsymbol{p}}_{curr.}^{T} \lfloor {}^{H,key}\boldsymbol{t}_{\times} \rfloor {}^{H,key}\overline{\boldsymbol{p}}_{key} = 0. \qquad (11) $$

There are only three up-to-scale entries in the translation-only essential matrix $\boldsymbol{E}_{H,key} = \lfloor {}^{H,key}\boldsymbol{t}_{\times} \rfloor$ in Eq. (11). Therefore, ${}^{H,key}\hat{\boldsymbol{t}}$ can be solved by a minimum of two point correspondences, as in [21].

We solve ${}^{H,key}\hat{\boldsymbol{t}}$ from the inlier point pairs of the 5-point RANSAC. In order to calculate translation, corresponding point vector pairs with big enough angles are necessary. After rotating the inlier point vectors into the keyframe heading frame, the point vector pairs are checked. Only pairs with angles beyond a certain threshold are used. In our implementation, the threshold is 5 degrees. Note that Eq. (11) has two mirrored solutions. The true direction can be determined by triangulating features and choosing the solution with all the features in front of the camera. To be more efficient, we use the propagated ${}^{B_t}\boldsymbol{p}_{R_k}$ to determine the direction of ${}^{H,key}\hat{\boldsymbol{t}}$. The solution that has a smaller angle with the expression of ${}^{B_t}\boldsymbol{p}_{R_k}$ in the heading frame is selected.

## 4 EXPERIMENTAL RESULTS

We evaluate the proposed ego-motion estimator on the widely recognized EuRoC MAV dataset. This dataset is comprised of multiple sequences with varying lighting conditions, and a maximum flight speed of 2.3 m/s. The proposed approach uses synchronized left-camera images and IMU measurements.

### 4.1 Data Pre-processing

In the EuRoC dataset, the IMU does not coincide with the forward-right-down body frame definition. As a result, the accelerometer's $x$-axis and $y$-axis do not directly measure the drag force in the body horizontal plane. A body-IMU rotation matrix is needed to transfer the accelerometer's measurements to the body frame. We assume that the coordinate frame of the reflective markers mounted on the MAV approximately coincides with the body frame. Because the origin of our body frame is the same as the IMU frame, we use the sensor rotational extrinsic data to rotate IMU measurements and ground truth to the body frame.

The proposed estimator is initialized just before taking off to comply with the drag force model, which only stands when the MAV is in flight. Visual updating starts after the second keyframe is captured. The initial value of $k_d$ was set to -0.2 for all the sequences. This value is close to the least-square solution calculated by the ground-truth velocity of the *Machine Hall 05* (*MH05*) sequence.

### 4.2 Results and Discussion

The main evaluation results of the proposed estimator in terms of time efficiency and accuracy are listed in Table 1. The C++ code of this work is implemented based on the open-sourced code of [22]. The proposed estimator is compared with a state-of-the-art (SOTA) VIO MSCKF [2] implemented by the same open-sourced repository [22]. The top group of Table 1 (from ① to ③) shows the test results of the proposed estimator with three different settings in the vision front-end. The bottom group (④ and ⑤)) shows the outputs of MSCKF in two settings. Here we only compare with MSCKF since the gap in accuracy is big. The comparison of MSCKF with other VIO solutions can be found in [9].

The time consumption measurements shown in the second and the third columns of Table 1 were collected during the tests on *MH05* sequence of the EuRoC dataset, running on a laptop computer. The root-mean-square error (RMSE) of absolute translation error (ATE) is the metric of estimation

Table 1: Accuracy and time efficiency of the proposed method (top group, rows 1 to 3) and the baseline method (bottom group, rows 4 and 5). *V101* to *MH05* shown in the eleven columns from the right are the names of flight sequences of the EuRoC MAV dataset. The data below the sequence names show the root-mean-square errors (RMSE) of the absolute translation errors (ATEs) of the estimated trajectories. **Bold** represents the overall best and underline represents the best of the proposed method.

| ID | ATT[1] | AVT[2] | NFP[3] | V101 | V102 | V103 | V201 | V202 | V203 | MH01 | MH02 | MH03 | MH04 | MH05 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ① | **4.41** | **3.11** | 30 | 5.12 | 1.19 | 8.04 | 4.21 | 2.11 | 5.90 | 4.70 | 13.4 | 12.0 | 2.37 | 2.76 |
| ② | 7.25 | 5.81 | 30 | 1.26 | 1.17 | 3.11 | 2.38 | 1.48 | 2.91 | 3.60 | 13.2 | 11.2 | 2.12 | 1.56 |
| ③ | 12.9 | 11.5 | 250 | 2.04 | 1.76 | 5.37 | 4.56 | 2.80 | 7.76 | 13.5 | 1.67 | **10.4** | 2.37 | 1.35 |
| ④ | 7.93 | 3.82 | 51 | 0.16 | 0.13 | 0.12 | 245 | 0.13 | **0.16** | 38.0 | 5.20 | 130 | 2.35 | **1.00** |
| ⑤ | 17.4 | 7.58 | 199 | **0.09** | **0.09** | **0.11** | **0.12** | **0.10** | 0.20 | **0.34** | **0.24** | 58.5 | **0.65** | 1.54 |

[1] Average total time (ATT) consumption (millisecond) of processing after receiving a new frame.

[2] Average vision-related time (AVT) consumption (millisecond) in processing a single frame, including feature tracking, feature detection, and outlier rejection of feature points.

[3] Number of feature points (NFP). For the proposed approach (top group), the shown number is the number of feature points to detect in a new keyframe. For MSCKF (bottom group), the shown number is the average number of tracked points in each frame.



accuracy. The RMSEs of ATEs of the eleven flight sequences of the EuRoC dataset are shown in Table 1. The alignment of the estimated trajectory and the ground-truth trajectory has 4 degrees of freedom (yaw and 3-d translation). The calculation is conducted by [23].

The upper group of Table 1 (from ① to ③) shows the test results of the proposed estimator with three different settings in the vision front-end. For ① and ②, 30 feature points are detected in a keyframe, and a new keyframe is defined when the number of inlier points is below 10. For ③, 250 feature points are detected in a keyframe, and a new keyframe is defined when there are fewer than 60 inliers. ① corresponds to the estimator that uses images that are downsampled to half of the original resolution. And it does not have histogram equalization as image preprocessing. The rest settings (② and ③) and MSCKF (④ and ⑤) use images with original resolution and preprocessed by histogram equalization. Thus the vision processing time consumption of ① is much lower than ② while sacrificing the accuracy. It is an extreme case that minimizes visual processing. In the 5-point RANSAC of ①, ②, and ③, six point pairs are selected as a sample. The total number of random trials is set to six. If the highest inlier rate is more than 60%, this sample's inlier point pairs are used to calculate the essential matrix.

Comparing ② and ③, it is obvious that more feature points do not necessarily improve the accuracy of the proposed estimator. This is different from MSCKF. As shown by ④ and ⑤, ⑤ detects and tracks more points and has higher accuracy. The reason is that the vision update of MSCKF is triggered when a point loses tracking or leaves the field of view. Thus a bigger number of points means more times of updating. But for the proposed estimator, vision update happens for every frame and the visual measurement is calculated from all points tracked in a frame. In this case, fewer points do not necessarily lead to a less accurate essential matrix. On the contrary, a big number of tracked points with tracking noise would deteriorate the accuracy. So the proposed estimator is more suitable for applications where the computational power for visual processing is very limited and only a small number of points can be detected and tracked.

For ②, the estimated trajectories of *MH02* and *MH03* are less accurate than other sequences. The trajectory errors are mainly caused by the big estimation errors at the beginning of the sequences. For *MH02*, the ATE of the second half of the estimated trajectory is 3.243. For *MH03*, after deleting 20% of the estimated trajectory from the beginning of the sequence, the ATE of the rest part of the trajectory is 2.979. From Fig. 4, we can see that the estimation converges to be accurate after around 30 seconds since taking off. So in real-world applications, it is better to have safety measures for the possible big drifts at the beginning of a flight, for example, a space without obstacles close by.

From Table 1, we notice that, in the sequences *V201*, *MH01*, and *MH03*, MSCKF ④ drifts after very slow motion, which leads to big trajectory errors. But in general, the proposed approach is less accurate than MSCKF. Despite that, as shown in Fig. 3, Fig. 4, and Fig. 5, the accuracy is acceptable for short-term ego-motion estimation. The advantages of the proposed estimator are time efficiency and robustness. As shown in Table 1, ① and ② consume less time than MSCKF ④. Another fact worth mentioning is that we observed in experiments that our 5-point RANSAC implementation runs slower than the OpenCV 8-point RANSAC adopted by MSCKF. It can be attributed to the highly optimized OpenCV library function. There can be a space to further shorten the time consumption by optimizing our implementation. As for robustness, the proposed estimator does not have big drifts as ④ when testing on the sequences *V201*, *MH01*, and *MH03*. As the proposed estimator does not map the environment, sudden loss tracking of many or all points

does not require re-initializing the system. In addition, the proposed estimator maintains accuracy even if there is not visual measurement anymore. As shown in Fig. 5, we cut off the camera image stream at the 52nd second after taking off, and the accuracy basically maintains. Obvious estimation errors are only observed in the velocity estimation of the $z$-axis in the last $\sim$12 seconds of flight.

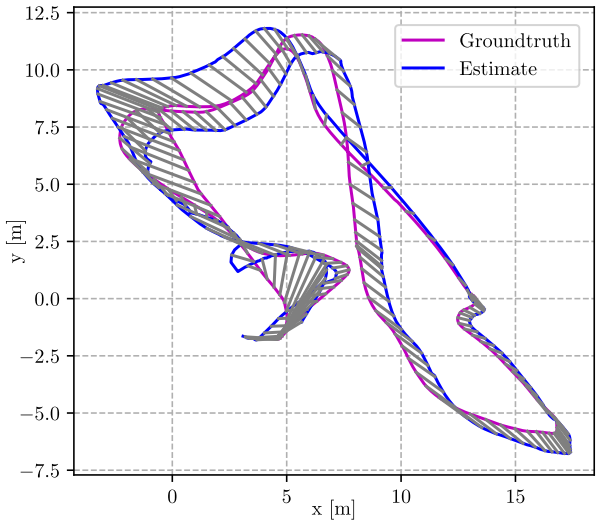
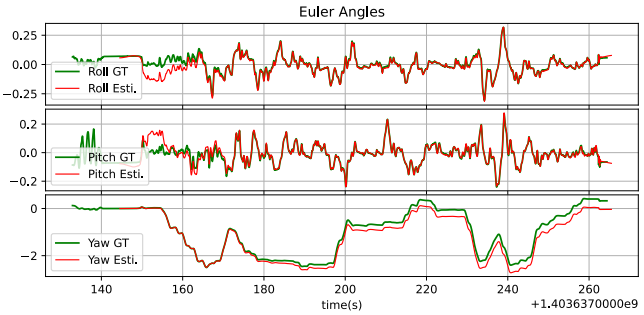
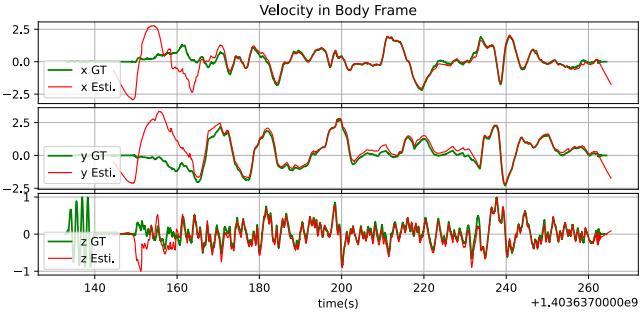



Figure 3: Estimated trajectory of *MH05* sequence by the proposed estimator ② in Table 1.

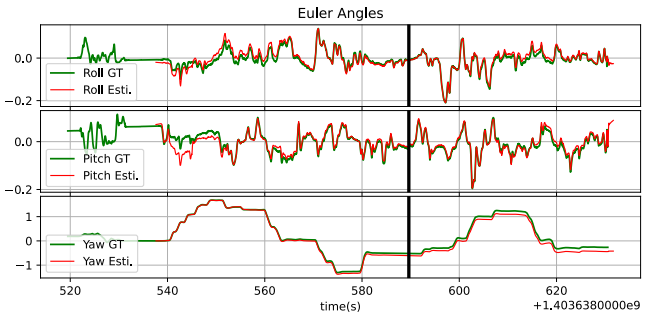



(a) Euler Angles Estimation (rad).

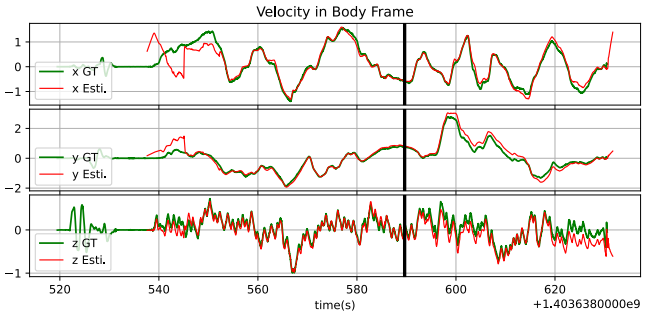



(b) Velocity Estimation Expressed in MAV Body Frame (m/s).

Figure 4: Estimated attitude and velocity of *MH03* sequence by the proposed estimator ① in Table 1.




(a) Euler Angles Estimation (rad).




(b) Velocity Estimation Expressed in MAV Body Frame (m/s).

Figure 5: Estimated attitude and velocity of *MH05* sequence by the proposed estimator ② in Table 1. Camera images are no longer used by the estimator after $\sim$590 seconds.

## 5 CONCLUSION AND FUTURE WORK

In this paper, we present a model-aided ego-motion estimator that is friendly to multirotor MAVs with limited processing power onboard. Minimized visual processing and a simple EKF-based backend that performs loosely-coupled visual-inertial fusion lead to high time efficiency. The proposed estimator is less accurate than a SOTA approach in comparison, but it can be enough for short-term navigation. Autonomous drone racing is a potential application. The proposed estimator's small requirement for processing power benefits other algorithms running onboard. It has acceptable drifts in estimating flight trajectories, which can be corrected by the racing gates that can serve as stationary landmarks. The estimation accuracy rarely drops in short term when the vision information becomes absent, implying that the estimator can stay functional when there is no valid feature point due to the motion blur in agile maneuvers.

## REFERENCES

[1] Shuo Li, Erik van der Horst, Philipp Duernay, Christophe De Wagter, and Guido C H E de Croon. Visual model-predictive localization for computationally efficient autonomous racing of a 72-gram drone. 2019. arXiv: 1905.10110.

[2] Mingyang Li and Anastasios I Mourikis. High-precision, consistent EKF-based visual-inertial odom-

etry. *The International Journal of Robotics Research*, 32(6):690–711, 2013.

[3] Tong Qin, Peiliang Li, and Shaojie Shen. VINS-Mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[4] Michael Bloesch, Michael Burri, Sammy Omari, Marco Hutter, and Roland Siegwart. Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback. *The International Journal of Robotics Research*, 36(10):1053–1072, 2017.

[5] Dinuka Abeywardena, Shoudong Huang, Ben Barnes, Gamini Dissanayake, and Sarath Kodagoda. Fast, onboard, model-aided visual-inertial odometry system for quadrotor micro aerial vehicles. *IEEE International Conference on Robotics and Automation*, pages 1530–1537, 2016.

[6] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *IEEE International Conference on Robotics and Automation*, pages 15–22, 2014.

[7] Stephan Weiss and Roland Siegwart. Real-time metric state estimation for modular vision-inertial systems. In *IEEE International Conference on Robotics and Automation*, pages 4531–4537, 2011.

[8] Matthias Faessler, Flavio Fontana, Christian Forster, Elias Mueggler, Matia Pizzoli, and Davide Scaramuzza. Autonomous, vision-based flight and live dense 3D mapping with a quadrotor micro aerial vehicle. *Journal of Field Robotics*, 33(4):431–450, 2016.

[9] Jeffrey Delmerico and Davide Scaramuzza. A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. *IEEE International Conference on Robotics and Automation*, pages 2502–2509, 2018.

[10] Dinuka Abeywardena, Sarath Kodagoda, Gamini Dissanayake, and Rohan Munasinghe. Improved state estimation in quadrotor mavs: A novel drift-free velocity estimator. *IEEE Robotics & Automation Magazine*, 20(4):32–39, 2013.

[11] Matthias Faessler, Antonio Franchi, and Davide Scaramuzza. Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *IEEE Robotics and Automation Letters*, 3(2):620–626, 2017.

[12] James Svacha, Giuseppe Loianno, and Vijay Kumar. Inertial yaw-independent velocity and attitude estimation for high-speed quadrotor flight. *IEEE Robotics and Automation Letters*, 4(2):1109–1116, 2019.

[13] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016.

[14] Marc Rigter, Benjamin Morrell, Robert G Reid, Gene B Merewether, Theodore Tzanetos, Vinay Rajur, KC Wong, and Larry H Matthies. An autonomous quadrotor system for robust high-speed flight through cluttered environments without GPS. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5227–5234, 2019.

[15] Zheng Huai and Guoquan Huang. Robocentric visual-inertial odometry. *The International Journal of Robotics Research*, 41(7):667–689, 2022.

[16] Joan Sola. Quaternion kinematics for the error-state kalman filter. *arXiv preprint arXiv:1711.02508*, 2017.

[17] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. *International Journal of Computer Vision Technical Report*, 1991.

[18] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, pages 430–443, 2006.

[19] H Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, 1981.

[20] Friedrich Fraundorfer, Petri Tanskanen, and Marc Pollefeys. A minimal case solution to the calibrated relative pose problem for the case of two known orientation angles. In *European Conference on Computer Vision*, pages 269–282. Springer, 2010.

[21] Chiara Troiani, Agostino Martinelli, Christian Laugier, and Davide Scaramuzza. 2-Point-based outlier rejection for camera-IMU systems with applications to micro aerial vehicles. In *IEEE International Conference on Robotics and Automation*, pages 5530–5536, 2014.

[22] Patrick Geneva, Kevin Eckenhoff, Woosik Lee, Yulin Yang, and Guoquan Huang. OpenVINS: A research platform for visual-inertial estimation. In *IEEE International Conference on Robotics and Automation*, 2020.

[23] Zichao Zhang and Davide Scaramuzza. A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7244–7251. IEEE, 2018.