A direct optic flow-based strategy for inverse flight altitude estimation with monocular vision and IMU measurements

A direct optic flow-based strategy for inverse flight altitude estimation with monocular vision and IMU measurements

Pakpong Chirarattananon*

Abstract

With tiny and limited nervous systems, insects demonstrate a remarkable ability to fly through complex environments. Optic flow has been identified to play a crucial role in regulating flight conditions and navigation in flies and bees. In robotics, optic flow has been widely studied thanks to the low computational requirements. However, with only monocular visual information, optic flow is inherently devoid of a scale factor required for estimating the absolute distance. In this paper, we propose a strategy for estimating the flight altitude of a flying robot with a ventral camera by combining the optic flow with measurements from an inertial measurement unit. Instead of using the prevalent feature-based approach for calculation of optic flow, we implement a direct method that evaluates the flow information via image gradients. We show that the direct approach notably simplifies the computation steps compared to the feature-based method. When combined with an extended Kalman filter for fusion of IMU measurements, the flight altitude can be estimated in real time. We carried out extensive flight tests in different settings. Among 31 hovering and vertical flights near the altitude of 40 cm, we achieved the RMS errors in the altitude estimate of 2.51 cm. Further analysis of factors that affect the quality of the flow and the distance estimate is also provided.

1 Introduction

In recent years, we have witnessed a dramatic growth of research and deployment of Micro Aerial Vehicles (MAVs) owing to numerous foreseeable applications. At decimeter scale, multi-rotor systems have thrived thanks to its mechanical simplicity and high maneuverability. Yet, small flying robots are still severely limited in terms of payload capabilities and computational power [1, 2, 3]. There remain several key challenges associated with the energetic cost of staying airborne, autonomous navigation through unstructured environments, etc., that prevent more pervasive uses of these small robots [4, 5]. These issues pose considerable complications towards the goal of autonomous operations. Fortunately, for several tasks related to sensing, localization, and navigation, biological systems provide us further inspirations for tackling the mentioned challenges.

With poor resolution compound eyes and only tiny nervous systems, insects demonstrate exceptional aerodynamic maneuvers such as grazing landing and agile collision avoidance. Scientists have discovered that birds and insects frequently use optic flow in short range navigation [6], flight speed regulation [7], landing [8], as well as obstacle avoidance [9]. Instead of directly measuring the distance, the optic flow technique measures the image motion, providing the ratio of flight speed to the distance to the surface [10]. Inspired by insects, researchers have pioneered the use of optic flow for robotics applications, aiming to leverage the low computational requirement and reducing the sensory components, which are crucial considerations for small aerial vehicles such as a 100-mg flapping-wing robot [11] or a 39-gram pocket-sized quadrotor [12].

In robotics, optic flow-based algorithms have been proposed and implemented in mobile robots and aerial robots for guidance and navigation [13]. In [14], Fuller and Murray devised an insect-inspired controller for detecting patterns of optic flow for a mobile robot with flight-like dynamics to navigate a corridor. Using the concept of time-to-contact, Izzo and de Croon introduced a strategy for landing on a surface using ventral optic flow based on the expansion of imaged ground [15]. In [3], a 2.6-g omnidirectional vision sensor enabled

*Pakpong Chirarattananon is with the Department of Mechanical and Biomedical Engineering, City University of Hong Kong, Hong Kong SAR, China (email: pakpong.c@cityu.edu.hk).

a 30-g miniature coaxial helicopter to control flight speed and stabilize the heading using wide-field optic flow. In addition to this, several optical devices have been developed to facilitate the use of insect-inspired strategies in robots. Examples include a miniature artificial compound eye with a panoramic field of view [16], a robust minimalistic high frame rate optic flow sensor [17], an open hardware camera with a CMOS image sensor for optic flow estimates designed for MAVs [18], and a 33-mg 1-D optic flow sensor for a flying microrobot [11].

Recently, researchers have explored the integration of optic flow information with other known quantities [19] or measurements from other sensors to exploit the "scaling" information, allowing true velocity or distance to be estimated [20, 18, 21, 22]. In [20], the opto-aeronautic algorithm was developed for estimation of wind speed and flight height using inputs from optic flow and air velocity sensors, whereas in [18], an ultrasonic sensor was incorporated. In other instances, inertial measurement units (IMUs) are chosen as they often exist for attitude stabilization. The fusion of measurements from two sensors usually requires a robust efficient strategy. In [21, 23], extended Kalman filters (EKF) were employed to recover a scale factor. In [22], the authors formulated a nonlinear estimation scheme for the task, allowing the transient response of the observer to be tuned. Based on the knowledge of control inputs, optic flow was used to robustly estimate the distance for landing in [19].

Several researchers have also demonstrated other novel methods that use optic flow in aid of the distance estimation or landing tasks. By imposing the control to ensure a constant time-of-flight and measuring the acceleration, the distance can be estimated with the approached proposed in [24]. Using only the knowledge of the control inputs and monitoring the oscillations of the robot without any knowledge of the robot's acceleration, the optic flow-based flight controller was shown capable of providing the estimate of flight altitude in hover as well as during landing [25].

Compared to popular alternative keyframe-based visual-SLAM methods that involve building and maintaining a map of visual features [26, 27, 28], these optic-flow based approaches are potentially less computationally intensive and more robust against failures as they do not need prolonged continuous feature tracking [21, 22].

Thus far, the majority of research involving combining optic flow and inertial measurements calculate the flow by detecting and tracking features between consecutive image frames using an established Lucas-Kanade (LK) algorithm [9, 25, 22]. The LK method is suitable for computing optic flow for a sparse feature set [29]. It was suggested that the required feature extraction and matching steps occupy most of the processing time, up to 85% in the case of [21]. Alternative to the LK method, there exists an alternative optic flow method developed by Horn *et. al.* [30], which directly provides time-to-contact information from image gradients without the need to identify and track image features. This principle has been extended to estimate the time-to-contact (without resolving for a scale factor) for motion control of mobile robots in [31].

In this paper, we aim to exploit this direct optic flow method in the context of flight altitude estimation. To achieve that, we consider a flying robot with a downward-looking monocular camera. We outline a theoretical basis to relate the motion of the robot and its altitude to the optic flow from the camera. Detailed analysis is included to identify factors that affect the performance of the computed flow. Then we propose an estimation routine based on a Kalman filter framework to incorporates the measurements from IMU to provide the scale factor needed to estimate the absolute distance. Static and flight experiments were performed to verify the models and test the estimation method. Further detailed analysis of the results is then discussed. In summary, our key contributions are threefold: (i) The modeling and implementation of the direct optic flow method in the context of a flying robot with a monocular vision. That is, we offer a model relating the expected optic flow to the motion of a robot flying over a flat terrain; (ii) Analysis of camera settings and flight parameters that have critical consequences to the accuracy of the optic flow data; (iii) An extended Kalman filter for estimating the inverse of flight altitude with a known observability condition. While alternative implementations with have been previously shown [24, 22], our method is simple, highly accurate at low altitude and robust against vibrations from flight.

The next section begins with a brief introduction of optic flow and the descriptions on how the LK method and the direct approach can be applied to deduce the motion (up to a scale factor) of a flying robot with a ventral vision. The analysis includes the introduction of three dimensionless quantities that are relevant to the accuracy of the computed flow. Section 3 shows a formulation of the inverse flight altitude and other flow measurements in the state-space representation form suitable for an EKF. An analytical condition for observability is provided. In section 4, we perform two sets of experiments to demonstrate

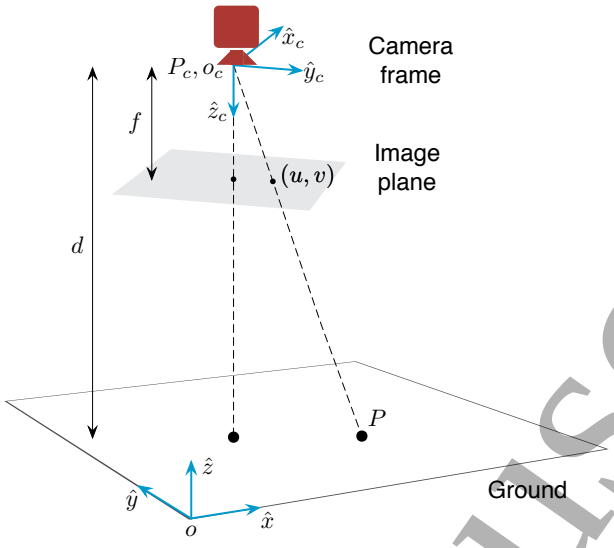

Figure 1: Definitions of the camera frame, the image plane, and the inertial frame.

the advantage of the direct optic flow approach over the LK method and to verify the importance of the introduced dimensionless quantities. Flight experiments are carried out in section 5 to examine our proposed estimation framework. Finally, conclusion and further discussions are provided in section 6.

2 Optic Flow

Suppose a stationary point of interest $P = \begin{bmatrix} X & Y & Z \end{bmatrix}^T$ in the inertial frame is projected onto the the location (u, v) on the camera plane as depicted in figure 1. Let $I(u, v, t)$ represents the intensity of an image pixel at this position at time t. Due to the movement of the camera, we expect the image of point P between two subsequent images to move from (u, v) to $(u + \Delta u, v + \Delta v)$. Under the *constant brightness assumption* or *optic flow constraint*, it is expected that $I(u, v, t) = I(u + \Delta u, v + \Delta v, t + \Delta t)$ [10]. To the first order approximation [32], it follows that

$$\frac{\partial I}{\partial u}\frac{\mathrm{d}u}{\mathrm{d}t} + \frac{\partial I}{\partial v}\frac{\mathrm{d}v}{\mathrm{d}t} + \frac{\partial I}{\partial t} = 0, \tag{1}$$

where $\mathrm{d}u/\mathrm{d}t$ and $\mathrm{d}v/\mathrm{d}t$ are components of the optic flow field.

2.1 Optic flow and camera motion

To relate the motion of a camera with respect to the inertial frame to the resultant optic flow, we consider a camera with a focal length f located at the point $P_c = \begin{bmatrix} X_c & Y_c & Z_c \end{bmatrix}^T$ defined in the inertial frame $o - \hat{x}\hat{y}\hat{z}$ as shown in figure 1. Let the coordinate frame associated to the camera be $o_c - \hat{x}_c\hat{y}_c\hat{z}_c$. The rotation matrix $R = \begin{bmatrix} \hat{x}_c & \hat{y}_c & \hat{z}_c \end{bmatrix}$ relates the orientation of the inertial frame and the camera frame. The image plane is perpendicular to the camera axis $(\hat{z}_c)$ at the distance f (focal length) away from the origin o_c. According to this arrangement, the point P is projected onto the image plane at the position (u, v) given by

$$\begin{aligned} u &= f\frac{R^T(P - P_c)\cdot e_1}{R^T(P - P_c)\cdot e_3} = f\frac{\hat{x}_c^T(P - P_c)}{\hat{z}_c^T(P - P_c)}, \\ v &= f\frac{R^T(P - P_c)\cdot e_2}{R^T(P - P_c)\cdot e_3} = f\frac{\hat{y}_c^T(P - P_c)}{\hat{z}_c^T(P - P_c)}, \end{aligned} \tag{2}$$

where e_i's are basis vectors (e.g., $e_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$).

If we denote the rotational rate of the frame o_c with respect to itself as $\omega = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$, the time derivatives of the camera frame axes become $\dot{\hat{x}}_c = \omega_z\hat{y} - \omega_y\hat{z}_c$, $\dot{\hat{y}}_c = \omega_x\hat{z} - \omega_z\hat{x}_c$, and $\dot{\hat{z}}_c = \omega_y\hat{x} - \omega_x\hat{y}_c$.

Assuming that point P is stationary with respect to the inertial frame ($\dot{P} = 0$), it can be shown that the time derivatives of the image location (u, v), or optic flow, are

$$\begin{aligned}\frac{\mathrm{du}}{\mathrm{d}t} &= -\frac{\hat{x}_c^T \dot{P}_c}{\hat{z}_c^T (P - P_c)} f + \frac{\hat{z}_c^T \dot{P}_c}{\hat{z}_c^T (P - P_c)} u + \omega_z v - \omega_y \frac{f^2 + u^2}{f} + \omega_x \frac{uv}{f} \\ \frac{\mathrm{dv}}{\mathrm{d}t} &= -\frac{\hat{y}_c^T \dot{P}_c}{\hat{z}_c^T (P - P_c)} f + \frac{\hat{z}_c^T \dot{P}_c}{\hat{z}_c^T (P - P_c)} v - \omega_z u + \omega_x \frac{f^2 + v^2}{f} - \omega_y \frac{uv}{f}.\end{aligned} \tag{3}$$

From this point, we consider the scenario where the camera is pointing towards the ground such that the camera axis is almost vertical. Define d as the distance from the origin of the camera frame to the ground plane along the camera axis, it turns out that the projection of point P along the camera axis, or $\hat{z}_c^T (P - P_c)$, is

$$\hat{z}_c^T (P - P_c) = d \left(1 - \frac{\frac{R_{31}}{R_{33}} \frac{u}{f} + \frac{R_{32}}{R_{33}} \frac{v}{f}}{1 + \frac{R_{31}}{R_{33}} \frac{u}{f} + \frac{R_{32}}{R_{33}} \frac{v}{f}} \right). \tag{4}$$

Furthermore, in the case that the camera axis is almost vertical, such as a downward-looking camera attached to a hovering quadrotor, we have $R_{31}, R_{32} \ll R_{33}$. Keeping only the first order terms (see [33] for a complete derivation), equation (4) together with equation (3) simplify to

$$\begin{aligned}\frac{\mathrm{du}}{\mathrm{d}t} &= -(\vartheta_x + \omega_y) f + \left(\vartheta_z - \vartheta_x \frac{R_{31}}{R_{33}} \right) u + \left(\vartheta_z \frac{R_{31}}{R_{33}} - \omega_y \right) \frac{u^2}{f} - \left(\vartheta_x \frac{R_{32}}{R_{33}} - \omega_z \right) v + \left(\vartheta_z \frac{R_{32}}{R_{33}} + \omega_x \right) \frac{uv}{f}, \\ \frac{\mathrm{dv}}{\mathrm{d}t} &= -(\vartheta_y - \omega_x) f + \left(\vartheta_z - \vartheta_y \frac{R_{32}}{R_{33}} \right) v + \left(\vartheta_z \frac{R_{32}}{R_{33}} + \omega_x \right) \frac{v^2}{f} - \left(\vartheta_y \frac{R_{31}}{R_{33}} + \omega_z \right) u + \left(\vartheta_z \frac{R_{31}}{R_{33}} - \omega_y \right) \frac{uv}{f},\end{aligned} \tag{5}$$

where we have introduced $\vartheta_x = \hat{x}_c^T \dot{P}_c / d$, $\vartheta_y = \hat{y}_c^T \dot{P}_c / d$ and $\vartheta_z = \hat{z}_c^T \dot{P}_c / d$. These quantities, which from now on will be referred to as *visual observables* [25] (also defined as apparent egovelocity in [20]). They provide velocity information of the camera frame up to a scale factor (d).

2.2 Lucas-Kanade optic flow algorithm

Equation (5) relates the optic flow $(\mathrm{d}u/\mathrm{d}t, \mathrm{d}v/\mathrm{d}t)$ to the motion and attitude of the camera. A number of previous works use feature extraction methods to initially detect features on the image and then employ the Lucas-Kanade tracker (LK) [34, 32] to determine the optic flow between consecutive image frames [13, 23, 22, 25, 24]. To estimate the visual observables with the *LK method*, we simplify equation (5) by neglecting $\vartheta_x \frac{R_{31}}{R_{33}} u$ and $\vartheta_y \frac{R_{32}}{R_{33}} v$ terms under the assumption that $R_{31}, R_{32} \ll R_{33}$. For each tracked feature $\Psi_i = [u_i, v_i]^T$, it satisfies

$$\Psi_i = \frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} u_i \\ v_i \end{bmatrix} \approx \underbrace{\begin{bmatrix} -f & 0 \\ 0 & -f \\ u_i & v_i \\ u_i^2/f & u_i v_i / f \\ u_i v_i / f & v_i^2 / f \\ -v_i & 0 \\ 0 & -u_i \end{bmatrix}^T}_{\Lambda_i} \underbrace{\begin{bmatrix} \vartheta_x + \omega_y \\ \vartheta_y - \omega_x \\ \vartheta_z \\ \vartheta_z R_{31}/R_{33} - \omega_y \\ \vartheta_z R_{32}/R_{33} + \omega_x \\ \vartheta_x R_{32}/R_{33} - \omega_z \\ \vartheta_y R_{31}/R_{33} + \omega_z \end{bmatrix}}_{\mathbf{b}}, \tag{6}$$

where we have defined a 2×7 matrix Λ_i and a vector of unknowns $\mathbf{b}$. With multiple features, we can construct $\Psi = \begin{bmatrix} \Psi_1 & \Psi_2 & \dots \end{bmatrix}^T$, $\mathbf{\Lambda} = \begin{bmatrix} \Lambda_1 & \Lambda_2 & \dots \end{bmatrix}^T$ and express the equation as a linear least-squares problem: $\Psi = \mathbf{\Lambda b}$ to solve for $\mathbf{b}$. The approach enables us to efficiently compute, for example, $\vartheta_x + \omega_y$, $\vartheta_y - \omega_x$, and ϑ_z using the optic flow data from various locations on the image. With the knowledge of the angular velocities from the IMU measurements, ϑ_x and ϑ_y are readily obtained.

2.3 Direct optic flow method

Instead of relying on feature detection and tracking method, it has been shown in [30, 31] that we can directly combine the optic flow constraint in equation (1) with the equation describing the camera motion (in the scenario described in figure 1, this corresponds to equation 5). If we let I_{ij} denote the pixel intensity at $I(u_i, v_j, t)$, the result directly relates the image gradients to the visual observables and the camera's attitude:

$$\underbrace{-\frac{\partial I_{ij}}{\partial t}}_{\Upsilon_{ij}} = \underbrace{\begin{bmatrix} -(\partial I_{ij}/\partial u) f \\ -(\partial I_{ij}/\partial v) f \\ (\partial I_{ij}/\partial u) u + (\partial I_{ij}/\partial v) v \\ (\partial I_{ij}/\partial u) u^2/f + (\partial I_{ij}/\partial v) uv/f \\ (\partial I_{ij}/\partial u) uv/f + (\partial I_{ij}/\partial v) v^2/f \\ -(\partial I_{ij}/\partial u) v \\ -(\partial I_{ij}/\partial v) u \end{bmatrix}^T}_{\chi_{ij}} \underbrace{\begin{bmatrix} \vartheta_x + \omega_y \\ \vartheta_y - \omega_x \\ \vartheta_z \\ \vartheta_z R_{31}/R_{33} - \omega_y \\ \vartheta_z R_{32}/R_{33} + \omega_x \\ \vartheta_x R_{32}/R_{33} - \omega_z \\ \vartheta_y R_{31}/R_{33} + \omega_z \end{bmatrix}}_{\mathbf{b}}, \tag{7}$$

where we have defined a scalar Υ_{ij}, a 1×7 matrix χ_{ij}, and a vector of unknowns $\mathbf{b}$. Once again, we have neglected two terms: $\vartheta_x \frac{R_{31}}{R_{33}} u$ and $\vartheta_y \frac{R_{32}}{R_{33}} v$ to arrive at equation (7). Similar to equation (6), this equation can be directly solved for $\mathbf{b}$ using the least-squares method in the form $\Upsilon = \chi \mathbf{b}$ (with $\Upsilon = \begin{bmatrix} \Upsilon_{11} & \Upsilon_{12} & \dots \end{bmatrix}^T$ and $\chi = \begin{bmatrix} \chi_{11} & \chi_{12} & \dots \end{bmatrix}^T$) using the values of I_{ij}'s from consecutive image frames. In contrast to the LK method, this *direct method* does not require the feature tracking process. Featureless areas in the image produce pixels with the values of $\partial I/\partial t$, $\partial I/\partial u$, and $\partial I/\partial v$ close to zero. Consequently, these points do not invalidate equation (7). This significantly reduces the computational requirement of the process.

2.4 Effects of pixelation, fame rate, and feature size on the accuracy of visual observables

In this section, we inspect the effects of camera pixelation, feature size, and frame rate on the accuracy of the optic flow estimate. To understand the relationship between different parameters, we consider a simplified situation where a hypothetical one-dimensional camera (the camera produces a one-dimensional array of pixels instead of the typical two-dimensional arrays) resides in a two-dimensional plane ($o - \hat{x}\hat{z}$). This setting can be regarded as a simplified version of the setup in figure 1.

2.4.1 Horizontal motion

Suppose the camera stays at a constant distance d from the ground, but it traverses laterally (along the $\hat{x}_c$-axis) at speed v_x. Furthermore, let the camera $\hat{x}_c$-axis coincides with the $\hat{x}$-axis of the inertial frame. and L be a length that describes the dominant lengthscale of the pattern on the floor. Assuming a uniform ideal lighting condition and a purely sinusoidal pattern on the floor, the brightness of the point $P(x)$ can be written as $\frac{I_0}{2}\left[\sin\left(2\pi\frac{x}{L}\right) + 1\right]$. Without loss of generality, the pixel intensity at point u on the image can be expressed as

$$I(u,t) = \frac{I_0}{2}\left[\sin\left(2\pi\left(\frac{ud}{fL} + \frac{v_x t}{L}\right)\right) + 1\right], \tag{8}$$

where f is the camera's focal length. Notice that the pixel brightness varies overtime as the camera moves at speed v in the direction parallel to the ground. In principle, to calculate the visual observable ϑ_x according to equation (7), we obtain $\partial I/\partial t$ and $\partial I/\partial u$ as

$$\begin{aligned} \frac{\partial I}{\partial t} &= \frac{2\pi v}{L}\frac{I_0}{2}\cos\left(2\pi\left(\frac{ud}{fL} + \frac{v_x t}{L}\right)\right) \\ \frac{\partial I}{\partial u} &= \frac{2\pi d}{fL}\frac{I_0}{2}\cos\left(2\pi\left(\frac{ud}{fL} + \frac{v_x t}{L}\right)\right), \end{aligned} \tag{9}$$

and ϑ_x is found to be

$$\vartheta_x = \frac{1}{f}\frac{\partial I}{\partial t}\Big/\frac{\partial I}{\partial u} = \frac{v_x}{d}, \tag{10}$$

as anticipated.

In practice, however, the camera sensor has a finite pixel size. If we let w denote the width of each pixel, the image brightness of a pixel located at u is averaged over the distance w such that

$$\begin{aligned}\bar{I}(u,t) &= \frac{1}{w}\int_{u-w/2}^{u+w/2} I(u,t)\,du \\ &= \frac{I_0}{2}\left[\frac{fL}{\pi dw}\sin\left(\frac{\pi dw}{fL}\right)\sin\left(2\pi\left(\frac{ud}{fL}+\frac{vt}{L}\right)\right)+1\right].\end{aligned} \tag{11}$$

Here, we introduce the notation $\delta[\cdot]$ to represent the difference between the ideal value and the practical value of a quantity of interest. The difference between $I(u,t)$ and $\bar{I}(u,t)$, or $\delta[I(u,t)]$, is the consequence of pixelation. We find that

$$\delta[I(u,t)] = \bar{I}(u,t) - I(u,t) = \left[\frac{fL}{\pi dw}\sin\left(\frac{\pi dw}{fL}\right)-1\right]\sin\left(2\pi\left(\frac{ud}{fL}+\frac{vt}{L}\right)\right). \tag{12}$$

To numerically evaluate the time and spatial derivatives of the pixel value for the calculation of visual observables, we consider the central difference:

$$\begin{aligned}\frac{\Delta}{\Delta t}\bar{I}(u,t) &= \frac{f_p}{2}\left(\bar{I}\left(u,t+\frac{1}{f_p}\right)-\bar{I}\left(u,t-\frac{1}{f_p}\right)\right), \\ \frac{\Delta}{\Delta u}\bar{I}(u,t) &= \frac{1}{2w}\left(\bar{I}(u+w,t)-\bar{I}(u-w,t)\right).\end{aligned} \tag{13}$$

where f_p is the camera's frame rate. Based on the previous definition of $\delta[\cdot]$, it follows that $\delta[\partial I/\partial t] = \frac{\Delta}{\Delta t}\bar{I}(u,t) - \partial I/\partial t$ and $\delta[\partial I/\partial u] = \frac{\Delta}{\Delta u}\bar{I}(u,t) - \partial I/\partial u$. After some algebraic manipulation, it can be shown that

$$\begin{aligned}\delta\left[\frac{\partial I}{\partial t}\right]\Big/\frac{\partial I}{\partial t} &= \left(\frac{fL}{\pi dw}\sin\left(\frac{\pi dw}{fL}\right)\right)\left(\frac{f_pL}{2\pi v}\sin\left(\frac{2\pi v_x}{f_pL}\right)\right)-1, \\ \delta\left[\frac{\partial I}{\partial u}\right]\Big/\frac{\partial I}{\partial u} &= \left(\frac{fL}{\pi dw}\sin\left(\frac{\pi dw}{fL}\right)\right)\left(\frac{fL}{dw}\sin\left(\frac{dw}{fL}\right)\right)-1.\end{aligned} \tag{14}$$

From equations (12) and (14), it can be seen that the numerical errors of $\partial I/\partial t$ and $\partial I/\partial t$ can be quantified using two dimensionless quantities $\pi dw/fL$ and $2\pi v/f_pL$. For the sake of simplicity, we define $\gamma_d = \pi dw/fL$ and $\gamma_x = 2\pi v_x/f_pL$. Physically, γ_d represents the ratio of the pixel size with respect to the feature size as expressed in the image frame, whereas γ_x is the ratio of the camera movement to the feature size between consecutive frames. Both quantities in equation (14) approach zero as $\gamma_d, \gamma_x \to 0$. Subsequently, we can approximate the error in the calculated visual observable due to the pixelation and numerical differentiation as

$$\left(\delta[\vartheta_x]/\vartheta_x\right)^2 \approx \left(\delta\left[\frac{\partial I}{\partial t}\right]\Big/\frac{\partial I}{\partial t}\right)^2 + \left(\delta\left[\frac{\partial I}{\partial u}\right]\Big/\frac{\partial I}{\partial u}\right)^2. \tag{15}$$

For illustrative purposes, we consider the case where $\gamma_d, \gamma_x \to 0$. It follows that equation (15) simplifies to

$$\left(\delta[\vartheta_x]/\vartheta_x\right)^2 \approx \left(\frac{\gamma_d^2}{6}+\frac{\gamma_x^2}{6}\right)^2 + \left(\frac{\gamma_d^2}{6}\left(1+\frac{1}{\pi^2}\right)\right)^2. \tag{16}$$

This suggests that both γ_d and γ_x directly affects the estimation of ϑ_x. We anticipate the accuracy of the estimate of ϑ_x to deteriorate as, for example, the feature size becomes too small compared with the altitude, or the camera moves excessively fast relative to the frame rate.

2.4.2 Vertical motion

In this case, we assume the camera, with the focal length f, is in an identical situation to the previous scenario, but it traverses vertically instead of horizontally. The expression for an image intensity is similar to equation (8):

$$I(u,z,t) = \frac{I_0}{2}\left[\sin\left(2\pi\frac{uz}{fL}\right)+1\right], \tag{17}$$

where $z = z(t)$ such that $\dot{z}(t) = v_z$. Partially differentiating equation (17) with respect to t and u yields the solution for the corresponding visual observables according to equation (7) as $\vartheta_z = -(\partial I/\partial t)/(u\partial I/\partial u)$. The effects of pixelation and sampling, nevertheless, only allow us to obtain the estimates of $\partial I/\partial t$ and $u\partial I/\partial u$ through the numerical differentiation in a similar fashion to equation (13). Employing the same method, it can be shown that

$$\begin{aligned}\delta\left[\frac{\partial I}{\partial t}\right]/\frac{\partial I}{\partial t} &\approx \left(\frac{fL}{\pi dw}\sin\left(\frac{\pi dw}{fL}\right)\right)\left(\frac{f_p fL}{2\pi u v_z}\sin\left(\frac{2\pi u v_z}{f_p fL}\right)\right) - 1,\\ \delta\left[\frac{\partial I}{\partial u}\right]/\frac{\partial I}{\partial u} &= \left(\frac{fL}{\pi dw}\sin\left(\frac{\pi dw}{fL}\right)\right)\left(\frac{fL}{dw}\sin\left(\frac{dw}{fL}\right)\right) - 1,\end{aligned} \tag{18}$$

where d is the instantaneous height of the camera, i.e., $z(t) = d(t)$. Notice that the expression of $\delta\left[\frac{\partial I}{\partial t}\right]/\frac{\partial I}{\partial t}$ in equation (18) is distinct from the one in equation (14) owing to the presence of u. To further understand the discrepancy brought by the discretization, we define $u^* = \max u$, as the maximum value of u, which is limited by the camera sensor's size, and $\gamma_z = 2\pi v_z u^*/f_p fL$. In the limit that $\gamma_d, \gamma_z \to 0$, we find that

$$\begin{aligned}\left(\delta[\vartheta_z]/\vartheta_z\right)^2 &\approx \left(\frac{\gamma_d^2}{6} + \frac{1}{6}\left(\frac{u}{u^*}\gamma_z\right)^2\right)^2 + \left(\frac{\gamma_d^2}{6}\left(1+\frac{1}{\pi^2}\right)\right)^2\\ &\leq \left(\frac{\gamma_d^2}{6} + \frac{\gamma_z^2}{6}\right)^2 + \left(\frac{\gamma_d^2}{6}\left(1+\frac{1}{\pi^2}\right)\right)^2.\end{aligned} \tag{19}$$

The uncertainty of ϑ_z has the same structure as that of ϑ_x in equation (16). However, the value of $\gamma_d = \pi dw/fL$, which is a function of the altitude, is now time-varying.

3 Estimation of Inverse Distance

In contrast to stereo vision, information from monocular vision such as optic flow or visual velocity is inherently devoid of a scale factor. The scale ambiguity can be resolved by fusing the visual observable from optic flow with acceleration measurements from the IMU. To begin, we evaluate the time derivative of d from the definition $(P_c + d\hat{z}_c)\cdot\mathbf{e}_3$ in section 2.1 as

$$\dot{d} = -\left(\vartheta_z + (\vartheta_x + \omega_y)\frac{R_{31}}{R_{33}} + (\vartheta_y - \omega_x)\frac{R_{32}}{R_{33}}\right)d. \tag{20}$$

Define $\alpha = d^{-1}$ as the approximate inverse distance of the camera from the ground, with the assumption that $\hat{z}$ and $\hat{z}_c$ are almost parallel ($R_{31}, R_{32} \ll R_{33}$), equation (20) becomes

$$\dot{\alpha} \approx \vartheta_z \alpha. \tag{21}$$

Subsequently, we obtain the time derivative of the visual observables according to the definitions given in section 2.1:

$$\begin{aligned}\dot{\vartheta}_x &\approx \omega_z\vartheta_y - \omega_y\vartheta_z + \frac{\hat{x}_c^T\ddot{P}_c}{d} + \vartheta_x\vartheta_z\\ \dot{\vartheta}_y &\approx \omega_x\vartheta_z - \omega_z\vartheta_x + \frac{\hat{y}_c^T\ddot{P}_c}{d} + \vartheta_y\vartheta_z\\ \dot{\vartheta}_z &\approx \omega_y\vartheta_x - \omega_x\vartheta_y + \frac{\hat{z}_c^T\ddot{P}_c}{d} + \vartheta_z^2.\end{aligned} \tag{22}$$

Notice that the terms with $\ddot{P}_c$ represent the acceleration of the camera in the camera frame. In the circumstance where the camera and an IMU are part of the same object, these terms are equivalent to the readings from the accelerometers (a_x, a_y, and a_z) combined with the projected gravity (where the angles, also used for flight attitude control, are obtained from fusion algorithms such as [35]). For instance, we have $\hat{x}_c^T\ddot{P}_c = a_x - R_{31}g$, where a_x is the the reading from the $\hat{x}$ axis of the accelerometer and R_{31} is approximately the pitch angle of the robot as internally estimated by the IMU. Treating $\mathbf{X} = \begin{bmatrix}\alpha & \vartheta_x & \vartheta_y & \vartheta_z\end{bmatrix}^T$

as a state variable, its dynamics are inherently nonlinear. Nevertheless, we can still express $\dot{\mathbf{X}}$ in the form resembling the state-space representation:

$$\underbrace{\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\alpha\\ \vartheta_x\\ \vartheta_y\\ \vartheta_z\end{bmatrix}}_{\dot{\mathbf{X}}(t)} = \underbrace{\begin{bmatrix}\vartheta_z & 0 & 0 & 0\\ a_x - R_{31}g & \vartheta_z & \omega_z & -\omega_y\\ a_y - R_{32}g & -\omega_z & \vartheta_z & \omega_x\\ a_z - R_{33}g & \omega_y & -\omega_x & \vartheta_z\end{bmatrix}}_{F(t)} \underbrace{\begin{bmatrix}\alpha\\ \vartheta_x\\ \vartheta_y\\ \vartheta_z\end{bmatrix}}_{\mathbf{X}(t)}, \tag{23}$$

where $F(t)$ functions as a state matrix. The corresponding output vector of the system $\mathbf{Y}$ contains the visual observables as estimated by the algorithm in section 2:

$$\begin{aligned}\mathbf{Y} &= \begin{bmatrix}\vartheta_x & \vartheta_y & \vartheta_z\end{bmatrix}^T\\ &= \underbrace{\begin{bmatrix}\mathbf{0}_{3\times 1} & I_{3\times 3}\end{bmatrix}}_{C}\mathbf{X},\end{aligned} \tag{24}$$

where C is an output matrix. Together, equations (23) and (24) describe the nonlinear dynamics of the state variable in a state-space representation form .

3.1 Extended Kalman filter

The dynamics of the system described by equations (22) and (20) are nonlinear in nature. While we can directly calculate the visual observables directly from the captured images and IMU measurements, the distance d, or its inverse α, is not directly measured. In theory, an observer can be designed to estimate α. In the case of a linear system, Kalman filter can be employed to recursively find an optimal estimate of the state vector. Nonlinear systems similar to our proposed system typically requires nonlinear observers or other variants of Kalman filters such as extended Kalman filter or unscented Kalman filter. By expressing the dynamics of the state vector in the standard state-space representation as in (23) and (24), it allows us to promptly estimate the state vector using the extended Kalman filter. This approach significantly simplifies the computation as well as the hardware implementation while providing a decent performance.

To employ the Kalman framework, we discretize the system in equations (23) and (24) using the forward Euler method [36]. Let T be a sample time and k be a time index such that $\mathbf{X}_{k+1} - \mathbf{X}_k \approx \dot{\mathbf{X}}(t_k)\,T = F(t_k)\,T\mathbf{X}(t_k)$. The discrete-time state-space form of the system is

$$\mathbf{X}_{k+1} = \underbrace{(I + F_k T)}_{A_k}\mathbf{X}_k + \mu_k, \quad \mathbf{Y}_k = C\mathbf{X}_k + \nu_k, \tag{25}$$

where we define A_k as a discrete-time state matrix, and μ_k and ν_k as 4×1 process noise and 3×1 observation noise assumed to be drawn from a zero mean multivariate normal distribution with covariance Q_k and R_k: $\mu_k \sim N(0, Q_k)$ and $\nu_k \sim N(\mathbf{0}, R_k)$. Equation (25) represents the system model. The update equation for the state estimate $\hat{\mathbf{X}}_k$ is then

$$\hat{\mathbf{X}}_{k+1} = A_k\hat{\mathbf{X}}_k + K_{k+1}\left(\mathbf{Y}_{k+1} - C_k A_k \hat{\mathbf{X}}_k\right),$$

with K_k being the optimal Kalman gain as described in [37]. The scheme allows us to iteratively obtain the estimate of $\mathbf{X}_k$, which includes the inverse altitude, from the measurements of visual observables in $\mathbf{Y}_k$.

3.2 Observability

To ensure that the proposed filter can reliably estimate α, the discrete-time system described in equations (25) must be observable. Herein, we analyze the observability of the respective system by inspecting the observability Gramian of a simplified scenario.

For a time-varying discrete-time system in the form of equation (25), the k-step observability Gramian is defined as

$$Q_k = C_0^T C_0 + A_1^T C_1^T C_1 A_1 + \ldots + A_1^T \ldots A_{k-1}^T C_{k-1}^T C_{k-1} A_{k-1} \ldots A_1. \tag{26}$$

The system is observable over k steps if and only if Q_k is full rank, i.e., Q_k is nonsingular [38]. For the case of our system described by equation (25), the 2-step observability Gramian is $Q_2 = C^T C + A_1^T C^T C A_1$. Assuming minimal angular velocity ($\omega_x, \omega_y, \omega_z \approx 0$), the determinant of the observability Gramian is

$$\det Q_2 = T^2 \left[1 + (1 + \vartheta_z T)^2\right]^2 \left[(a_x - R_{31}g)^2 + (a_y - R_{32}g)^2 + (a_z - R_{33}g)^2\right]. \quad (27)$$

It can be seen that, for a finite sample time T, the system is observable as long as $(a_x - R_{31}g)^2 + (a_y - R_{32}g)^2 + (a_z - R_{33}g)^2 > 0$. In fact, each of these terms corresponds to the acceleration of the camera along its body axis. The summation is the square of the total acceleration. In other words, the system is observable as long as its net acceleration is not zero. In theory, this means we are unable to get the estimate of α if the camera is mounted on a robot in a perfect hovering condition. In practice, slight movement is likely adequate to ensure that the system is observable. This observability condition, however, does not indicate the degree of observability.

One measure of observability is the square root of the condition number of Q_k [39]. This condition number, κ, indicates how large the change in the initial condition in one direction affects the output in another direction. The task of finding the state vector from the observed outputs becomes ill-conditioned at excessively large condition number.

4 Performance Evaluation

In this section, we perform a set of experiments to compare the performance of the direct optic flow method to the feature-based method outlined in section (2). In addition, the effects of pixelation, fame rate, and feature size are investigated in terms of two parameters (γ_d and γ_f) introduced in section (2.4).

4.1 Flow calculation and experimental setup

The setup for the following experiment is illustrated in figure 2 (in the handheld experiment configuration). We use an oCam-5CRO-U camera (Hardkernel) and a single board computer Odroid-XU4 (Hardkernel) for capturing images, processing, and computing the visual observables. The camera is physically attached to an AscTec Hummingbird quadrotor (Ascending Technologies). The onboard computer communicates with the quadrotor via serial communication. This allows us to access the measurements from the IMU on the robot at the rate of ≈ 150 Hz. Command signals from the ground station for the robot are transmitted to the single board computer via UDP over wifi at 100 Hz.

We use a Python script for managing the communication between the onboard computer and other devices (the camera, robot, and ground station). Another Python routine is implemented to process images for visual observables. For the LK method, the pipeline starts by detecting 50 features in each image using the FAST corner detectors [40]. The features are matched to the next image with the Lucas–Kanade optical flow algorithm to directly provide $\mathrm{d}u/\mathrm{d}t$ and $\mathrm{d}v/\mathrm{d}t$ [34]. The corner detectors and LK algorithms were implemented in Python using OpenCV wrappers. The visual observables are then calculated from 50 points using equation (6) and the method of least squares. To eliminate the angular velocity components present in equation (6), the corresponding measurements from the IMU are used for subtraction.

The direct flow method involves the spatial and time derivative of pixel values, which can be susceptible to image noises. Therefore, each image is initially smoothened using a 5×5 average kernel. The spatial derivatives, $\partial I/\partial u$ and $\partial I/\partial v$, are computed using normalized $3{\times}3$ Sobel kernels in their respective directions. The time derivative is directly obtained by subtracting two consecutive smoothened frames. The visual observables are computed using the method of least squares based on equation (7), and the angular velocities are subtracted out by the IMU measurements. To further reduce the computational requirement, only one in every sixteen pixels (in a grid arrangement) was used in the least squares regression. This amounts to $4,800$ pixels out of the total of $76,800$ pixels for a 320×240 image. These computations were carried out in Python.

Moreover, for both methods, we may opt to apply equations (6) or (7) to fully estimate seven unknowns. Then only the first three elements of $\mathbf{b}$ are used for the EKF, discarding other elements. Instead of using the *full* models, we may neglect the last four rows of Λ_i and χ_{ij} to estimate only the truncated $\mathbf{b}$ (which is

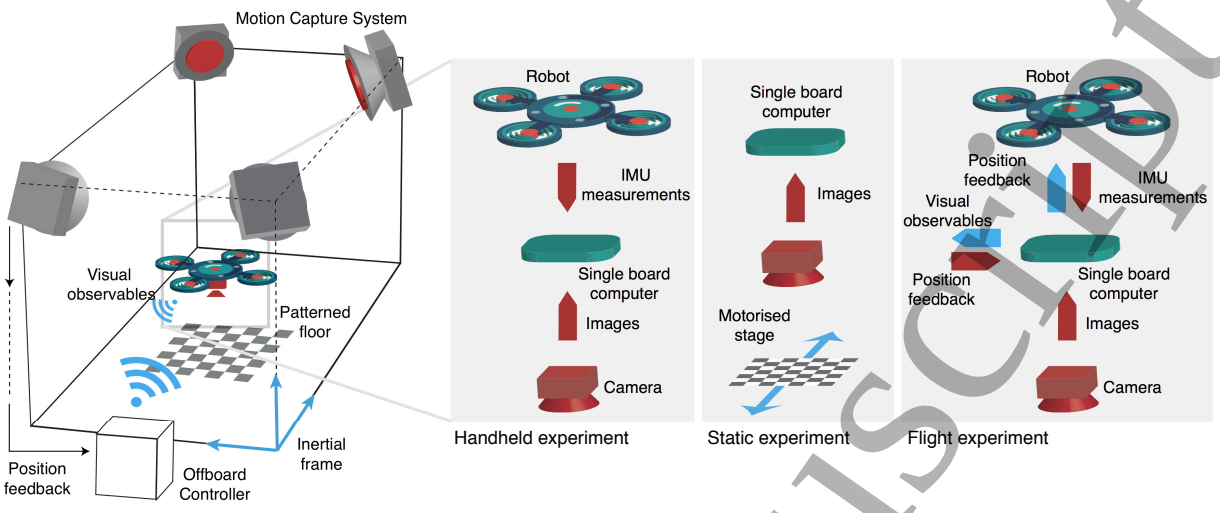

Figure 2: A schematic diagram depicting the setups for three types of experiments. The arena is affixed with a checkerboard pattern on the ground. The motion capture system provides ground truth measurements for a robot-camera platform. The single board computer functions differently for different experiments.

a 3×1 vector in this case). This approximation is reasonable due to the assumption that the camera axis is almost vertical ($R_{31}, R_{32} \ll R_{33}$) and the angular velocities are relatively small compared with the visual velocities. These *reduced* models potentially decrease the computational load.

The combined camera-computer-robot system was placed in the $3.0 \times 3.0 \times 2.5$ m arena fitted with six motion capture cameras (OptiTrack Prime 13w). The motion capture system tracks four retroreflective markers placed on the robot to provide real-time position and orientation feedback for the ground computer. This information can be processed to obtain the expected visual observables, acceleration, velocities, and angular velocities of the camera. Thanks to the accuracy (~ 0.1 mm) and sample rate (240 Hz) of the motion capture system, we treat these quantities as ground truth measurements for evaluating the performance of the estimated visual observables or IMU measurements.

4.2 Comparison between LK and direct methods

First, we aim to compare the visual observables calculated from the traditional LK method and the direct method. We placed a checkerboard pattern with the square size of 5×5 cm on the floor. The camera-robot prototype was commanded to take 320×240 images at 30 and 60 frames per second (fps) and store the images locally. Simultaneously, IMU measurements from the robot were logged. The communication between the onboard computer and the ground station allows us to synchronize and associate each image frame with the exact pose of the robot as determined by the motion capture system. The robot and the camera were handheld manually moved around at the distance ≈ 30 cm to 100 cm from the ground. Several image sequences were recorded, providing more than 2,500 images for each frame rate. The recorded data were then post-processed to compute the visual observables using the LK method (equation 6) and the direct method (equation 7), both with full and reduced models. Since the same sets of images were used for both methods, we can make a direct comparison of the results.

Figure 3 shows the experimental results of the calculated visual observables from the LK and direct methods using reduced and full models. The computed visual observables are compared with the ground truth values calculated from the camera trajectory provided by the motion capture system to obtain RMS errors for comparison. The results show that at 30 fps, both methods produced visual observables with similar RMS errors. The difference between full and reduced models are also insignificant. The accuracy of both approaches improves dramatically (the RMS errors are $\approx 60 - 70\%$ smaller) when the frame rate is 60 fps. The improvement is slightly more pronounced for the LK method. Overall, there is no significant difference between the direct method and the LK method in terms of RMS errors.

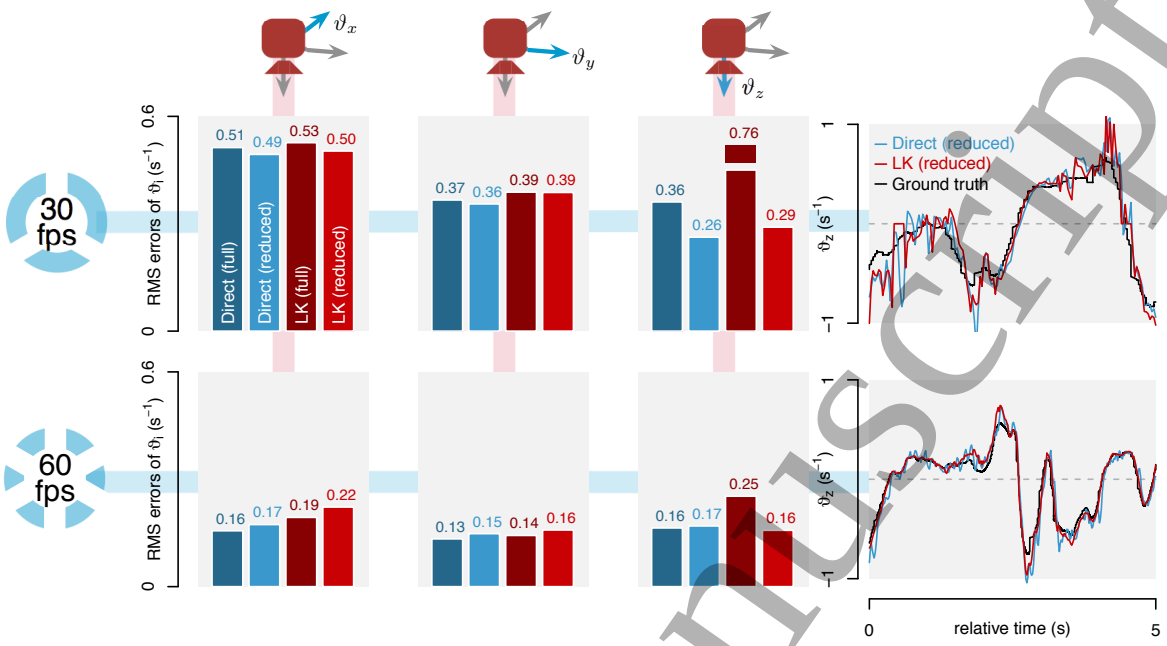

Figure 3: RMS errors of visual observables from the handheld experiments. (Left) Bar plots contrasting the RMS errors of the calculated visual observables obtained from the LK method and the direct method at two camera frame rates (30 fps and 60 fps). (Right) The plots showing 5-second portions of the visual observable along the camera axis calculated from both methods compared to the ground truth.

The time plot in figure 3 reveals a similar trend in more detail, compared to the ground truth, the computed ϑ_z's are subject to high frequency disturbances. With an increased frame rate from 30 to 60 fps (effectively reducing $\gamma_x, \gamma_y, \gamma_z$ by a factor of two), both methods provide smoother ϑ_i's that follow the ground truth more closely with less oscillations.

Furthermore, we compare the time the Python script used to produce the visual observables by both methods in full and reduced modes (including only the image processing time, i.e., the without capturing or image acquiring process). Figure 4 presents the distribution of the time taken per image frame. The data reveals that the median computational times per frame (solid horizontal lines) are approximately 10 ms, similar for all configurations. Nevertheless, it can be seen that the reduced direct method consistently used only $9.3 - 10.0$ ms ($25^{th} - 75^{th}$ percentile) per frame, whereas the reduced LK method required $1.7 - 22.3$ ms ($25^{th} - 75^{th}$ percentile) per frame. The noticeable small deviation in processing time of the direct method is due to its deterministic nature of the algorithm. In contrast, the speed of feature-based methods needs more computational time for complex images or when features are lacking. As a result, on average, the LK methods require slightly more computational time than the direct methods (presented as solid dots in Fig 4). The time used in image processing task is crucial as it was shown to occupy as much as 85% processing time in the estimation routine in [21]. The consistency in computation time is desirable for a real-time estimation process.

4.3 Comparison between different patterns

In the previous performance tests and subsequent experiments, we used a checkerboard pattern to provide visual texture for optic flow. The advantages of a checkerboard pattern are, for example, the presence of clear edges and corners and the lengthscale of the pattern can be precisely defined. However, it is possible that a particular pattern may affect the performance of the direct or LK methods differently.

To verify this hypothesis, we carried out further handheld experiments with identical conditions to the tests in section 4.2. In this experiments, however, the checkerboard pattern was replaced with two different patterns: connected rings and leaf as shown in Figure 5. The two new patterns differ from the original

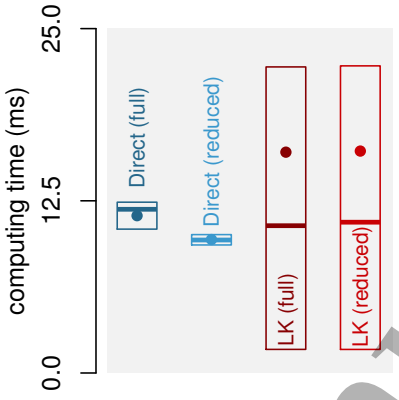

Figure 4: A box plot presenting the distribution of the times used to process one image frame using the direct and LK methods with full and reduced models. Each box shows the median, 1^{st} and 3^{rd} quartiles. The dots represent the mean computational times.

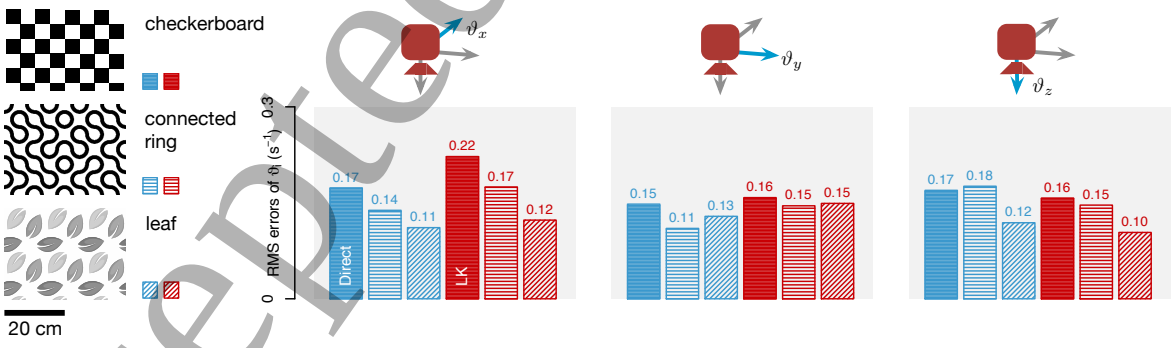

Figure 5: RMS errors of visual observables from the handheld experiments with different patterns. Two additional patterns with less structure (connected ring and leaf) were tested with the proposed direct method and LK method to evaluate the effects of texture on the performance of the methods.

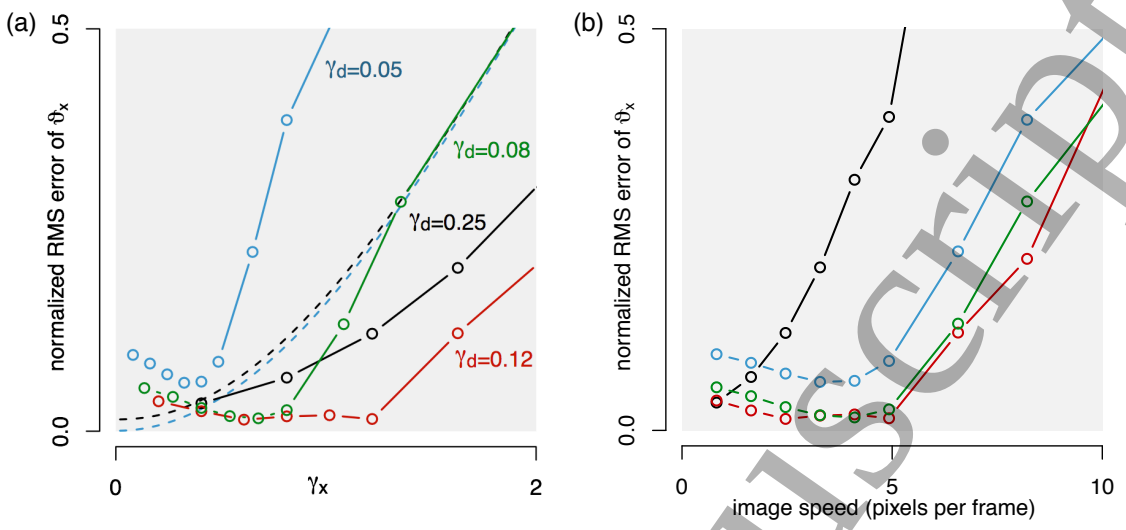

Figure 6: Normalized RMS errors of the visual observable from different pattern sizes and camera frame rates. (a) The normalized RMS errors of ϑ_x against the dimensionless parameter γ_x at four values of γ_d's. The dashed lines show the bounds of errors for $\gamma_d = 0.08$ (green) and $\gamma_d = 0.25$ (black) as theoretically predicted by equation (15). (b) The RMS errors of ϑ_x versus the image displacement (in pixel).

checkerboard pattern by the absence of straight edges. The connected ring pattern is also devoid of sharp corners. Figure 3 shows the results of the calculated visual observables from the LK and direct methods using the reduced models from images taken at 60 fps. The RMS errors of ϑ_i's from the two new patterns are consistently smaller than that of the checkerboard pattern. However, the improvement is marginal. The tests here suggest that both methods are reasonably robust to the texture used in indoor environments.

4.4 Characterization of pixelation, speed and feature size on the accuracy of visual observables

In section 2.4, we examined how the finite pixel size, camera frame rate, and relative motion affect the estimate of visual velocities under some simplifying assumptions. Three dimensionless quantities: γ_d, γ_x and γ_z, were found to play a role in the estimates of ϑ_x and ϑ_z. To validate our analysis for the case of a camera having some horizontal speed, we systematically carried out a static experiment with the details as follows.

An identical camera (oCam-5CRO-U) and a single board computer (Odroid XU4) to the previous experiment were statically mounted as shown in figure 2. A checkerboard pattern was affixed on a linear motorized stage such that the pattern was parallel to the image plane, approximately 50 cm away from the camera. Here, the checkerboard pattern approximately represents the sinusoidal pattern in the analysis in section 2.4. To elaborate, for a Checkerboard pattern with 5×5 cm squares, the texture repeats itself every 10 cm. The corresponding lengthscale L is, therefore, 10 cm (see Figure 5 for an example). The motorized stage allows the checkerboard pattern to programmatically travel along the camera's $\hat{x}_c$-axis. The motion of the pattern relative to the camera essentially provides equivalent effects to a moving camera pointing towards the fixed ground as previously assumed.

The camera was commanded to record images at 60 fps for 200 seconds. In the meantime, the stage moved the pattern back and forth at constant speed of approximately 3.8 cm.s^{-1} as measured by the motion capture system. This generates a constant ϑ_x. The experiment was repeated with the checkerboard patterns of different sizes, including 1×1, 2×2, 3×3, and 5×5 cm, to vary the values of γ_d without altering d. The stored images were then downsampled to simulate lower camera frame rates down to 1 fps. This results in different values of γ_x. We then calculated the visual observable (ϑ_x) of different cases using the reduced direct method. With the ground truth value of ϑ_x from the aid of the motion capture system, we obtained the RMS errors of ϑ_x from different conditions. The errors are shown in figure 6a in terms of γ_x and γ_d, along with the theoretically predicted bounds by the formula from section 2.4.1.

The trend seen in figure 6a generally agrees with the predictions that γ_x and γ_d have important roles in the accuracy of the visual observable estimates. That is, the errors grow as γ_x increases. For the same value of γ_x, $\gamma_d = 0.25$ produces greater errors than $\gamma_d = 0.12$. There, nevertheless, exist two anomalies. First, we observe relatively large errors in ϑ_x at small γ_x (~ 0.3 or smaller). Second, at very small γ_d (0.05 and 0.08), the errors are unexpectedly substantial, exceeding the predicted bounds.

To further understand the cause of the relatively large errors at low γ_x, we reproduce figure 6a to show the RMS errors as a function of the image velocity in the unit of pixels per frame (this quantity corresponds to $v_x f/wdf_p$). Apart from those with $\gamma_d = 0.25$, which have relatively large RMS errors owing to the the large γ_d, other sets of data suggest that the RMS errors in ϑ_x are minimized when the motion between image frames is just below 5 pixels. In other words, the errors grow as the motion drops below 5 pixels per frame. We speculate that the implementation of the 5×5 average kernel to smoothen image in the processing step prevents the detection of minuscule movements between two consecutive image frames, causing the rise in the RMS errors of ϑ_x.

Other factors may also contribute to the observed anomalies. For example, we used checkerboard patterns for the experiments to approximate the sinusoidal functions. The implementation of the algorithm to calculate visual velocities also deviates slightly from the theoretical method as the average filter and Sobel operators were used to deal with noisy images. The large errors from exceptionally small $\gamma_d = 0.05$ (from a 5×5 cm pattern) could be a result of relatively sparse data points as the majority of image pixels are black or white with no spatial or temporal variation.

5 Flight Experiments

To verify that the proposed inverse distance estimation framework can reliably produce the estimate of flight altitude despite the presence of noise from calculation of optic flow, evaluation of the time derivative of visual observables, and the use of accelerometer measurements, in this section, we perform flight experiments with a quadcopter in the indoor flight arena to investigate the performance of the proposed strategy under various flight conditions.

5.1 Flight tests

To perform flight experiments, we setup the arena and prepare the robot similar to the experiment in section 4.1 as depicted in figure 2. Here, the onboard computer processes the captured images for visual observables in real-time at 60 fps using the reduced direct flow method implemented on a Python script. The arena ground was fitted with a 5×5 cm checkerboard pattern. According to the characterization result in figure 6, we aimed to ensure that γ_x is less than 0.5 in flight to minimize the errors of computed visual observables. For a hypothetical horizontal flight speed of 0.5 m·s^{-1} and a 5×5 cm checkerboard pattern, γ_x is expected to be ≈ 0.5. In contrast, if a 1×1 cm pattern was chosen, γ_x would be excessively large at ≈ 2.5.

The image processing routine is identical to the handheld experiment in section 4.1. The single board computer is also responsible for retrieving IMU measurements and passing on the position feedback for the robot to ensure that it stays at the commanded position. The IMU measurements and computed visual observables are transmitted to the ground station via UDP. We found that the visual observable data consistently lagged behind the IMU measurements by ≈ 84 ms. This number represents the onboard image processing time, which can be compensated by delaying the IMU measurements by the same amount. The ground controller executes the estimation algorithm in real-time using the xPC target environment (MathWorks Matlab) at the rate of 500 Hz. The computation involved with the inverse distance estimation is relatively inexpensive compared to the calculation of visual observables. We opted to implement this part of the algorithm on the ground station owing to the ease of implementation and debugging purposes. In flight conditions, the single board computer generally reports the CPU usage of less than 10% on one of the eight available cores.

In the implementation of the Kalman estimator, in theory, the covariance matrices associated to the process noise and the measurement noise (Q_k and R_k) can be estimated from the current system state and empirical data from the IMU. For simplicity, we treated them as constant and they were experimentally tuned. The gyroscopic readings were used for the angular velocities, the accelerometer outputs were taken for a_x, a_y, and a_z needed for the matrix F in equation (23), and the pitch and roll angles from the IMU

fusion used for flight attitude stabilization were also used as R_{31} and R_{32} for the matrix F. Moreover, a 3^{rd}-order Butterworth low pass filter with a cutoff frequency of 6 Hz was applied to all IMU measurements and computed visual observables in order to minimize the measurement noise and vibration effects. To investigate the performance of the proposed estimation schemes, we carried out the flight tests with three flying patterns: hovering, vertical, and horizontal flights. First, the robot was commanded to hover at 40 cm altitude ($\gamma_d \approx 0.04$). Second, the robot was instructed to follow a sinusoidal altitude setpoint, centered at 40 cm with the amplitude of 10 cm, with the period ranging from 5 to 10 seconds ($\gamma_z < 0.01$). Third, we let the robot fly in a horizontal circular pattern with a radius of 30 cm at the constant altitude of 40 cm. The periods were set to values between 8 and 20 seconds ($\gamma_x, \gamma_y < 0.25$).

In total, we conducted 8 hovering flights, 23 vertical flights, and 23 horizontal flights. For each flight, we extracted the estimation data from the middle portion of the flights, excluding taking off and landing periods. Each flight provides $30 - 45$ seconds of valid data for us to examine the results.

5.2 Estimation results

Figure 7 shows the altitude estimation results (from inversion of the estimated α's) and relevant quantities from three flights selected to represent different flight modes (hovering, vertical and horizontal). The RMS errors in α and altitude (d) of these flights are 0.12, 0.08, 0.21 m^{-1} and 1.91, 1.48, 3.86 cm respectively. According to the plots, the estimate of α from the horizontal flight is visibly poorer than the others. In fact, the same trend is observed across several flights.

In terms of visual observables, the hovering flight produced relatively small visual observables in all directions. The vertical flight resulted in a noticeably greater ϑ_z, corresponding to the vertical flight speed of up to ≈ 12 cm·s^{-1}. On the other hand, the horizontal flight reached the maximum speed of ≈ 40 cm·s^{-1}, rendering ϑ_x and ϑ_y to periodically peak at $\approx \pm 1$ s^{-1}. When compared to the ground truth calculated from the motion capture measurements, however, the exists no significant correlation between the RMS errors of the visual velocities and their magnitudes. Figure 7 further suggests that, among these flight conditions, the robot did not deviate considerably from the upright orientation as the pitch and roll angles (R_{31}/R_{33} and R_{32}/R_{33}) remained bounded by 0.12 rad, or 7°.

Figure 9 further illustrates the altitude estimation errors from flights from three flight conditions over time. The plot verifies that the estimation errors do not tend to grow over time. Nevertheless, some temporal patterns can be observed in the case of vertical and horizontal flight. The respective oscillations are related to the periodicity of the commanded flight trajectories. Figure 9 also strengthens the observation of the significantly larger estimation errors in the case of horizontal flights.

To understand what contributes to the relatively large estimation errors seen in horizontal flights, we compiled the data from all 54 flights and plot the RMS errors in α and d against the RMS values of various quantities in figure 8a. Color codes are used to distinguish different flight types. The plots confirm the initial supposition that the horizontal flights suffered relatively large estimation errors. The majority of hovering and vertical flights have the RMS errors in d under 3 cm, whereas the horizontal flights resulted in the RMS errors ranging from $\approx 3 - 9$ cm, depending on the period of the circular trajectory. The averaged RMS errors in altitude is is 2.51 cm for hovering and vertical flights and 4.54 cm for horizontal flights.

5.3 Analysis of the results

In fact, figure 8a suggests that the magnitudes of ϑ_x and ϑ_y might play an important role in the quality of the inverse altitude estimates. The plots between the RMS errors in α and the RMS of other quantities, including pitch and roll angles, acceleration, and the condition number, do not show any visible trend. To quantify the correlation between the RMS errors of the estimate and these parameters, we calculated the correlation coefficients between the pairs, $\rho(\cdot, \cdot)$. The magnitudes of the correlation coefficients are presented in figure 8b, with 0 indicating no correlation and 1 indicating total correlation.

One may expect some relationship between the estimation errors and the pitch/roll angles owing to the assumption $R_{31}, R_{32} \ll R_{33}$ used in the derivation of the estimation method. In addition, the absence of acceleration ($\ddot{P}_c$) may violate the observability condition according to equation (26). Similarly, the square root of the condition number, $\kappa^{1/2}$, could have a similar role as a large condition number leads to a reduced

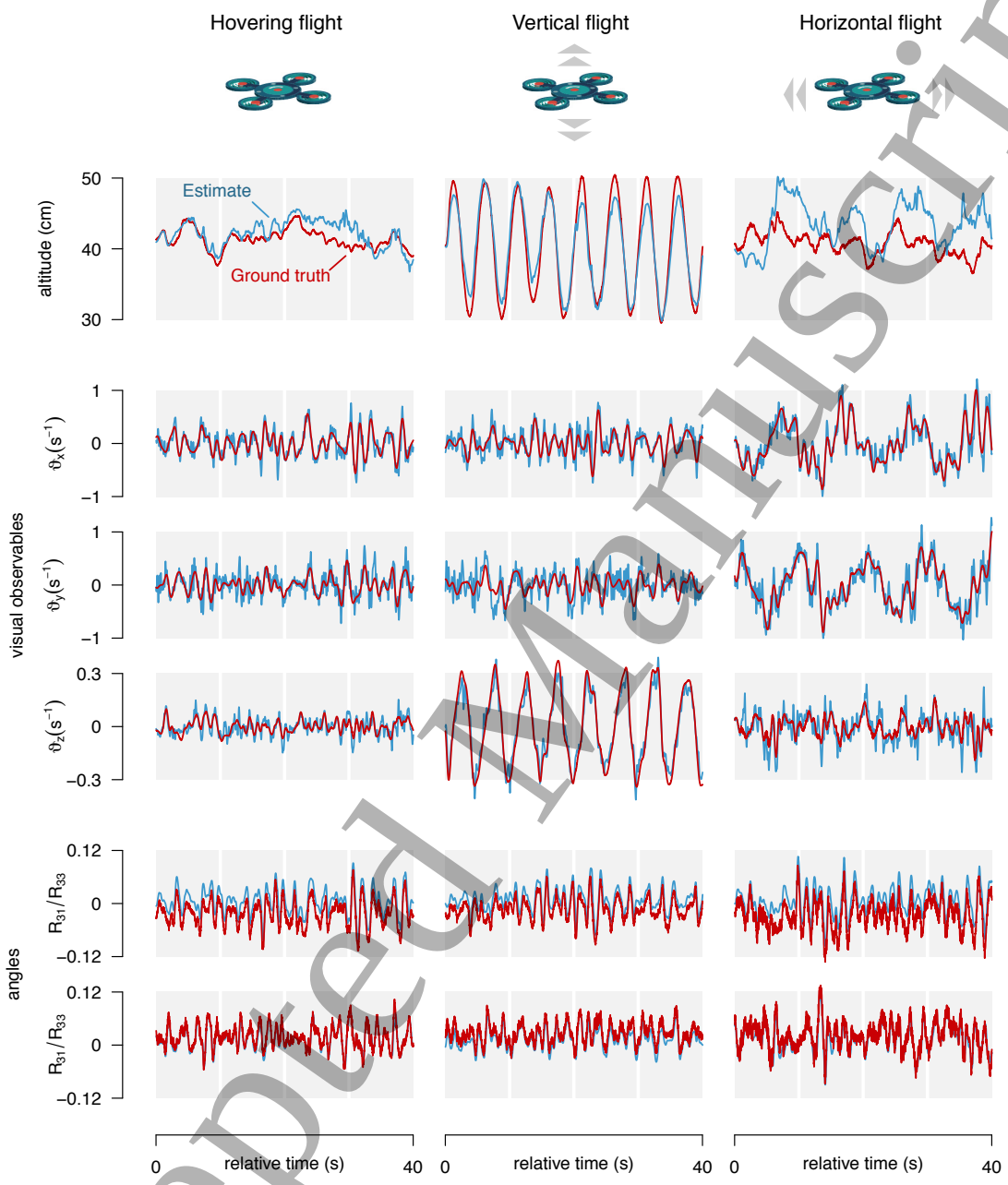

Figure 7: Measurements and estimates from example hovering, vertical, and horizontal flights. (Top) The plots show the estimated altitude compared to the ground truth from the motion capture system. (Middle) The visual observables along the camera frame axes calculated by the reduced direct method are compared with the ground truth. The difference characteristics of different flight modes are evident. (Bottom) The pitch and roll angles of the robot given by the IMU and the motion capture system. The data suggests that the attitude of the robot do not deviate significantly from the perfect hovering condition in all three flight modes.

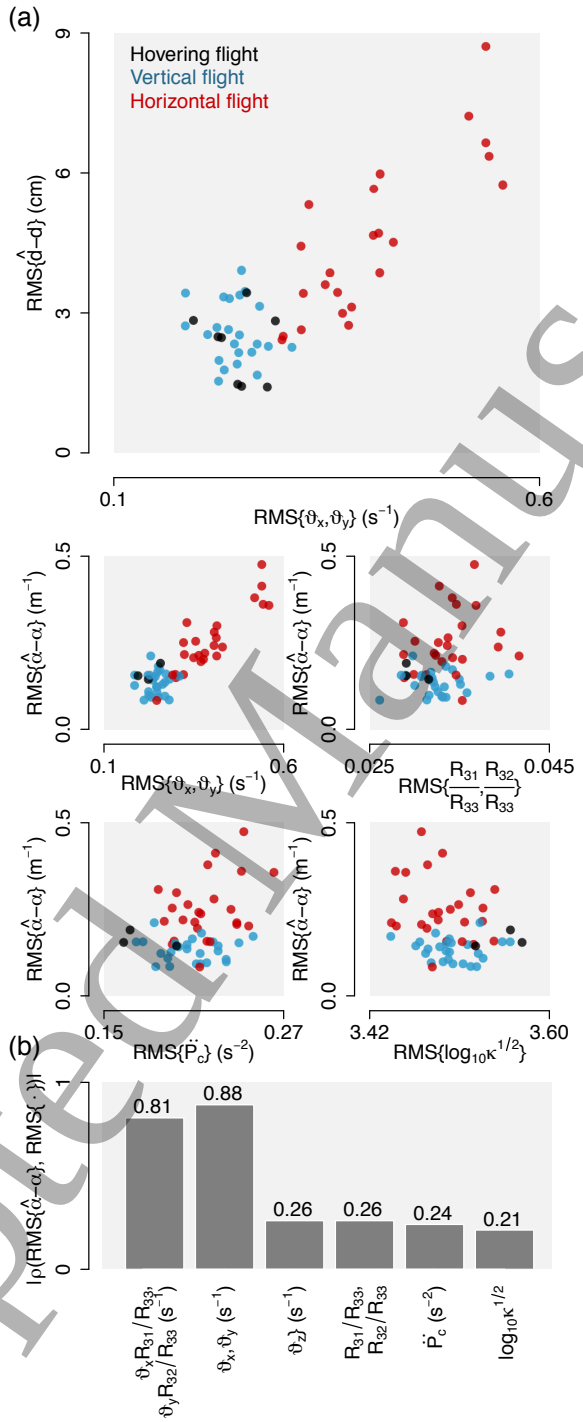

Figure 8: Averaged RMS errors of altitude ($\hat{d} - d$) and inverse altitude ($\hat{\alpha} - \alpha$) from all 54 flights plotted against the averaged RMS values of other flight parameters. The plots allow qualitative visual inspection of the correlation between the estimation errors and various quantities. (Bottom) Comparison between the correlation coefficients of the RMS errors in α and different quantities. The plot reveals a strong correlation between the altitude errors and the visual observables in the horizontal directions.

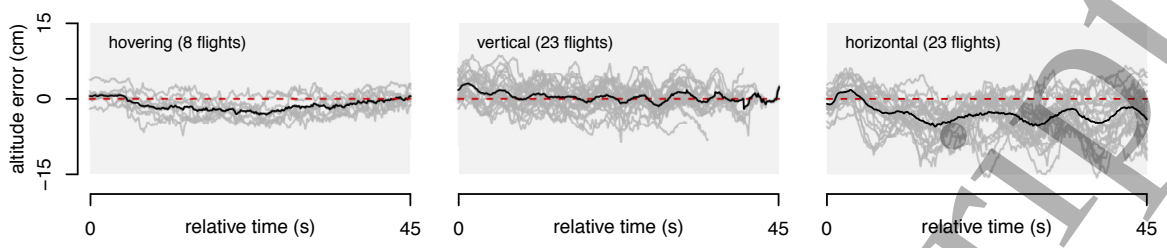

Figure 9: Altitude estimation error over time. The plots show the altitude estimation with respect to the groundtruth (dashed red) from all flights (solid grey) in three flight modes. The averaged errors are shown in solid black lines.

observability. However, figure 8b verifies that only ϑ_x, ϑ_y and $\vartheta_x R_{31}/R_{33}, \vartheta_y R_{32}/R_{33}$ have a significant correlation with the estimation errors, with the coefficients of 0.88 and 0.81.

A possible explanation for the observed phenomena is, for hovering flights or flights with low ground speeds, the variations in the pitch/roll angles and acceleration are insignificant. It can be seen in figure 8a that the RMS values of $R_{31}/R_{33}, R_{32}/R_{33}$ or $\ddot{P}_c$ of horizontal flights cannot be distinguished from those of hovering or vertical flights. The same observation applies to the condition number. This means that all flights are similar in attitude and degree of observability.

The likely explanation why the magnitudes of ϑ_x and ϑ_y have considerable effects on the quality of the estimates could be due to the approximation made to achieve equation (7). Therein, we assumed $\vartheta_z - \vartheta_x R_{31}/R_{33}, \vartheta_z - \vartheta_y R_{32}/R_{33} \approx \vartheta_z$ based on the assumption on attitude: $R_{31}, R_{32} \ll R_{33}$. The former condition becomes less accurate when $\vartheta_x, \vartheta_y \geqslant \vartheta_z$, which is expected for the case of horizontal flight with little variation in altitude. A similar approximation was also made to obtain equation (21) from equation (20). Therefore, for horizontal flights, we anticipate the algorithm to perform better without assuming $\vartheta_z - \vartheta_x R_{31}/R_{33}, \vartheta_z - \vartheta_y R_{32}/R_{33} \approx \vartheta_z$. This, nonetheless, will inevitably lead to increased complexity in the estimation routine.

6 Conclusion and Discussion

In this paper, we have tackled the problem of an online altitude estimation for flying robots with a monocular vision and an IMU using an optic flow-based approach. Compared to the traditional feature-based methods, we experimentally demonstrated that the performance of the direct method is comparable to that of the former method both in terms of accuracy and computational speed for providing the visual observables in our specific test conditions. The quality of the computed visual observables was also shown to depend on various factors related to the camera and flight configurations. We introduced three dimensionless parameters to use as guidelines for appropriately determining the camera settings under various flight conditions.

With the proposed EKF scheme, we successfully estimated the flight altitude with the average RMS error of 2.51 cm considering 31 hovering and vertical flights at ≈ 40 cm altitude. The quality of the estimate was found to deteriorate upon an increase in the horizontal components of the visual observables. This is likely owing to the approximations made in the calculation of the visual observables and the estimation scheme.

Our estimation results compare favorably with other prominent works. Among 31 flights with the average RMS errors of 2.51 cm, our best three flights have the RMS errors of 1.42, 1.43, and 1.48 cm. We believe one contribution to the relatively small errors we achieved is the computational efficiency of the direct optic flow approach. This, in turn, enables us to compute the visual observables at 60 fps, significantly improving the accuracy of the estimation. However, thus far, our work still critically relies on the assumption of flat ground with no camera occlusion. In contrast, in [22], a random sample consensus (RANSAC) algorithm was employed to reject outliers in optic flow measurements. The strategy improved the robustness of the estimation in more realistic, cluttered environments.

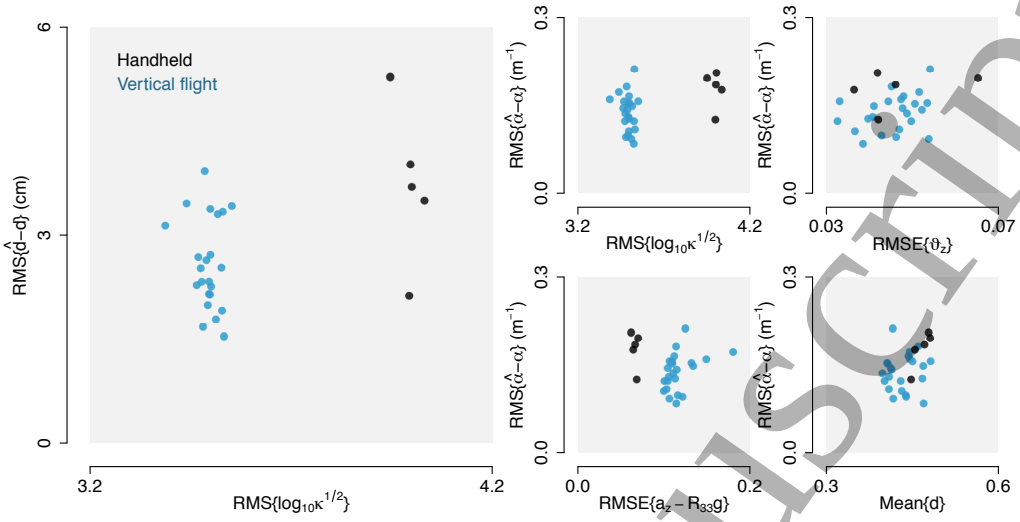

Figure 10: Averaged RMS errors of altitude ($\hat{d} - d$) and inverse altitude ($\hat{\alpha} - \alpha$) from vertical flights at 40 cm and simulated vertical flights from the handheld experiments. The plots show the RMS errors against other quantities to reflect the effects of vibration on the estimates.

6.1 Effects of vibration on altitude estimation

Unlike applications on mobile robots or controlled experiments [31, 24], in the context of flying robots, one major challenge in achieving accurate estimates of the inverse distance from the combination of IMU measurements and the images is owing to the present vibrations in flight. In evidence of this, in [21], it was reported that the RMS errors in velocity estimates in flight were higher than those from a handheld experiment, citing the vibrations and motion blur as the cause of inaccuracies. Similarly, in [22], the results showed that the RMS errors of the estimated velocity from flight are approximately one order of magnitude larger than the values obtained from simulated data or a handheld experiment.

To further inspect this claim, performed additional handheld experiments to imitate vertical flights in section 5. In this situation, the robot was manually moved vertically with the amplitude of ~ 10 cm at the altitude of ~ 40 cm, similar to the previous vertical flights. The camera captured series of images, computed the visual observables and the inverse altitude was estimated using the extended Kalman filter in a similar fashion. Five simulated flights were recorded. The estimation results and some parameters are presented with the previous vertical flight data in Figure 10. The figure also verifies that the handheld experiments were performed at the mean altitude of ~ 40 cm, comparable to previous vertical flights.

We found that the errors in altitude estimates from the handheld experiments are marginally larger than those from vertical flights. To explain the results, we found that, when compared to the ground truth, the quality of computed visual observables (ϑ_z) from the handheld experiments did not improve from the vertical flights (see Figure 10). This suggests that the mechanical vibration did not significantly affect the quality of the images. On the other hand, the vertical component of the acceleration from the IMU ($a_z - R_{33}g$ in equation (23)) from the handheld experiments were found to be more accurate than the ones from flights when compared to the ground truth. This alone is likely to adversely affect the altitude estimates.

The oscillation from flight, nevertheless, also brings about the more overall acceleration required for the observability of the inverse altitude as described by equation (27). In evidence of this, we compare the condition number of observability Gramian (Q_k) in the form of $\log_{10} \kappa^{1/2}$ as shown in Figure 10. It can be seen that condition numbers of handheld flights are consistently and noticeably higher than those from actual flights, indicating that handheld flights have relatively poor observability. This leads us to believe that the vibration from flight causes unfavorable effects on the IMU measurements, but simultaneously improve the observability of the estimation scheme.

While we experimentally found that the flight vibration can be beneficial to the estimates, there are

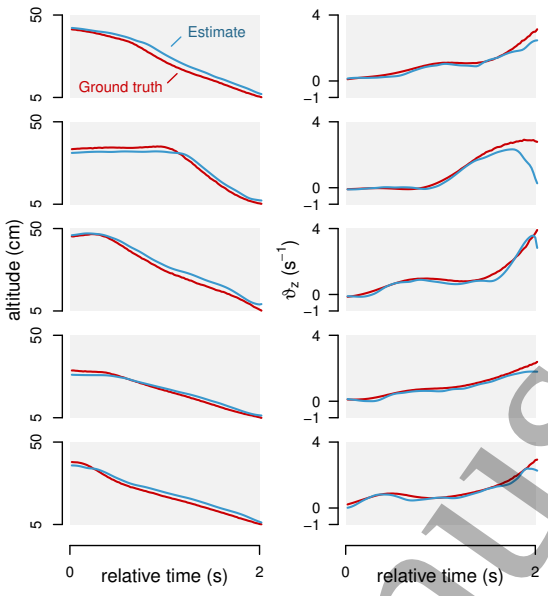

Figure 11: Altitude and associated visual observable during landing maneuvers of five representative flights. The plots show the flight trajectory over two seconds until the robot reached the altitude of 5 cm during landing. The averaged RMS errors in altitude estimates over this period of the five presented flights are 2.68, 2.18, 2.49, 1.43, and 2.06 cm. The values of ϑ_z can be seen rising rapidly towards the end of the landing as a result of the reduced altitude.

several experimental conditions that may affect the final outcomes. For instance, the degree of oscillations may vary between robots and flight controllers used. Therefore, we believe that our findings here should be interpreted with care.

6.2 Altitude estimation during landing maneuvers

Using the strategy borrowed from biology, flying robots have demonstrated an ability to achieve smooth landing by holding constant the time to contact with the surface [15, 41], similar to how honey bees control their speed by holding constant the rate of optic flow [31]. The technique enables insects and robot to automatically reduce the speed of flight upon reaching the contact without the knowledge of the actual distance to the surface.

With the estimate of distance, aerial robots potentially benefit from the ability to deploy a landing gear at a suitable moment or to approach the surface at arbitrary speed [42, 43]. That is, the landing task is no longer constrained to a constant rate of optic flow. To illustrate that our proposed strategy can reliably provide the distance estimate as the robot approaches the ground, we plot the landing trajectories from five representative flights in figure 11. This shows that the EKF accurately predicts the altitude at various landing speed with the RMS errors on the order of 2 cm. Upon approaching the surface at approximately constant speed, the respective visual observable ϑ_z rises radically and reaches the value of ≈ 4 s^{-1} in final moments (significantly higher than 0.3 s^{-1} observed in flights as shown in figure 7). The values of ϑ_z calculated from the direct method were highly accurate until the robot was ≈ 10 cm from the ground, at which point the field of view is likely too small for features to be captured by the camera.

6.3 Estimation errors at higher flight altitude

In the flight experiments, the robot was commanded to perform various flights near the altitude of 40 cm with the resultant RMS estimation errors of 2.51 cm. When it comes to landing flight with the flight altitude below 40 cm, we found that the RMS errors were generally below 2.50 cm. Because of the nonlinear dependence of the visual observables on the flight altitude d (for example, at very large d, $v_i's$ converge to zero for the same

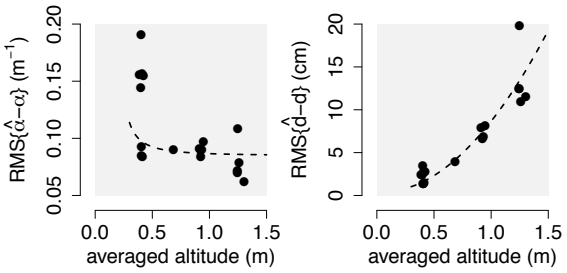

Figure 12: The plots showing the estimation errors of flight inverse altitude and altitudes from vertical flights performed at various height.

flight speed), it is reasonable to assume that the RMS errors of α or d are dependent on the flight altitude. The estimation is likely more accurate at lower altitude.

To demonstrate this, we performed ten more hovering flights at higher setpoint altitude ranging from $\sim$ 70 to 125 cm. The RMS errors in $\hat{\alpha}$ and $\hat{d}$ are plotted against the average flight altitude with previous hovering flight data included in Figure 12. The results suggest that as d_0 increases, the RMS errors of $\hat{\alpha}$ declines, but the estimation error of the flight altitude still increases. When the robot hovers at $\sim$ 1.2 m, the altitude estimation error is $\sim$ 10 cm. This is comparable to the results from in [22], where a robust feature-based ego-motion estimation algorithm was shown to provide the RMS error of 9.23 cm for a flight at 1 m altitude. Our results indicate the effectiveness of the proposed implementation of the direct optic flow method and the EKF scheme that are relatively simple in comparison.

To gain further insights into the relationship between the expected estimation errors from various flight altitude, we consider the steady-state condition of the Kalman estimation in a simplified situation. Assume that the robot is only moving vertically. The vertical trajectory oscillates about the nominal altitude d_0 with some speed $\hat{z}_c^T \dot{P}_c = \bar{v}_z$ and acceleration $a_z - R_{33}g = \bar{a}_z$. If we treat $\bar{v}_z$, $\bar{a}_z$ and d_0 to be approximately constant, the state matrix originally defined in equation (23) becomes

$$F(t) = \begin{bmatrix} \bar{v}_z/d_0 & 0 & 0 & 0 \\ 0 & \bar{v}_z/d_0 & 0 & 0 \\ 0 & 0 & \bar{v}_z/d_0 & 0 \\ \bar{a}_z & 0 & 0 & \bar{v}_z/d_0 \end{bmatrix}. \tag{28}$$

With this constant trajectory assumption, we assume further that the process noise and measurement noise are independent of d_0 and are also constant. In such situation, the extended Kalman filter predicts a steady-state covariance matrix $\bar{\Sigma}$ that can be evaluated from the Ricatti equation [37]

$$\bar{\Sigma} = A\bar{\Sigma}A^T + Q - A\bar{\Sigma}C^T \left(C\bar{\Sigma}C^T + R\right)^{-1} C\bar{\Sigma}A^T, \tag{29}$$

where $A = I + FT$ according to the discretization previously used in equation (25).

Up to this point, we have argued that, if the robot was following the same trajectory at different altitude, we anticipate that the only difference would be d_0 as the actual velocity and acceleration are independent of altitude. For the sake of simplicity, we also assume that the vertical velocity and acceleration are also constant. Doing so allows us to evaluate a steady-state covariance matrix of the state estimation. This covariance matrix will differ when d_0 varies. Hence, it allows us to understand the effect of d_0 on the anticipated altitude estimation error.

Using $\bar{v}_z = 0.2$ ms^{-1} and $\bar{a}_z = 0.2$ ms^{-1} as nominal conditions, we solve the Ricatti equation for $\bar{\Sigma}$. The expected variance of $\hat{\alpha}$ is given by the first element of $\bar{\Sigma}$. If we let $\tilde{\alpha}^2$ denotes this value, it is conceivable that $\tilde{\alpha}$ relates to the RMS error of $\hat{\alpha}$ from flight experiments. We find that $\tilde{\alpha}$ also reduces as d_0 increases as shown as a dashed line in Figure 12. In terms of the actual flight altitude, the fact that $\alpha = d^{-1}$ translates to $\tilde{\alpha} \sim \tilde{d}/d_0^2$. Therefore, we can also predict the uncertainty in $\hat{d}$ from $\tilde{\alpha}$. This prediction, $\tilde{d}$, is also shown in Figure 12 as a dashed line. The trend agrees with our experimental results that the estimation error of flight altitude grows as the altitude increases.

To further understand the simplified analysis here, it should be mentioned that the role of d_0 on the value of $\tilde{\alpha}$ stems from the diagonal element of F in equation (28). The characteristic of plots in Figure 12(b) is insensitive to simulation parameters. For instance, if the covariance matrix Q and R are scaled up by a factor of γ, Equation (29) shows that $\bar{\Sigma}$ will be scaled up by the same factor. That is, the shape of lines in Figure 12 are intact.

It is also important to note that, the trend predicted in Figure 12 is a result of the particular implementation of the extended Kalman filter we propose. It can be seen that the implementation here differs from previous works [22, 19]. The proposed scheme can be used whether the visual observables are obtained via the LK method or the direct method and the estimation errors are likely to depend on the altitude with a similar characteristic. Other implementation methods will likely result in a different relationship between the estimation error and altitude from our results in Figure 12. The quality of the altitude estimation depends not only on the accuracy of the visual observables but also on the estimation method.

7 Acknowledgement

This work was partially supported by the Research Grants Council of the Hong Kong Special Administrative Region of China (grant number CityU-21211315). The author would like to thank Sawyer B Fuller for an insightful discussion and comments in the preparation of the manuscript and the anonymous reviewers for their valuable comments and suggestions that substantially improve the quality of the paper.

References

[1] K. Y. Ma, P. Chirarattananon, S. B. Fuller, and R. J. Wood, "Controlled flight of a biologically inspired, insect-scale robot," *Science*, vol. 340, no. 6132, pp. 603–607, 2013.

[2] J. A. Roll, B. Cheng, and X. Deng, "An electromagnetic actuator for high-frequency flapping-wing microair vehicles," *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 400–414, 2015.

[3] R. J. Moore, K. Dantu, G. L. Barrows, and R. Nagpal, "Autonomous mav guidance with a lightweight omnidirectional vision sensor," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 3856–3861, IEEE, 2014.

[4] D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones," *Nature*, vol. 521, no. 7553, pp. 460–466, 2015.

[5] W. Shyy, C.-k. Kang, P. Chirarattananon, S. Ravi, and H. Liu, "Aerodynamics, sensing and control of insect-scale flapping-wing flight," in *Proc. R. Soc. A*, vol. 472, p. 20150712, The Royal Society, 2016.

[6] P. S. Bhagavatula, C. Claudianos, M. R. Ibbotson, and M. V. Srinivasan, "Optic flow cues guide flight in birds," *Current Biology*, vol. 21, no. 21, pp. 1794–1799, 2011.

[7] S. B. Fuller, A. D. Straw, M. Y. Peek, R. M. Murray, and M. H. Dickinson, "Flying drosophila stabilize their vision-based velocity controller by sensing wind with their antennae," *Proceedings of the National Academy of Sciences*, vol. 111, no. 13, pp. E1182–E1191, 2014.

[8] E. Baird, N. Boeddeker, M. R. Ibbotson, and M. V. Srinivasan, "A universal strategy for visually guided landing," *Proceedings of the National Academy of Sciences*, vol. 110, no. 46, pp. 18686–18691, 2013.

[9] F. Van Breugel and M. H. Dickinson, "The visual control of landing and obstacle avoidance in the fruit fly drosophila melanogaster," *Journal of Experimental Biology*, vol. 215, no. 11, pp. 1783–1798, 2012.

[10] S. S. Beauchemin and J. L. Barron, "The computation of optical flow," *ACM computing surveys (CSUR)*, vol. 27, no. 3, pp. 433–466, 1995.

[11] P.-E. J. Duhamel, C. O. Perez-Arancibia, G. L. Barrows, and R. J. Wood, "Biologically inspired optical-flow sensing for altitude control of flapping-wing microrobots," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 2, pp. 556–568, 2013.

[12] S. Mintchev, L. Daler, G. L'Eplattenier, L. Saint-Raymond, and D. Floreano, "Foldable and self-deployable pocket sized quadrotor," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 2190–2195, IEEE, 2015.

[13] H. Chao, Y. Gu, and M. Napolitano, "A survey of optical flow techniques for robotics navigation applications," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1-4, pp. 361–372, 2014.

[14] S. B. Fuller and R. M. Murray, "A hovercraft robot that uses insect-inspired visual autocorrelation for motion control in a corridor," in *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pp. 1474–1481, IEEE, 2011.

[15] D. Izzo and G. D. Croon, "Landing with time-to-contact and ventral optic flow estimates," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 4, pp. 1362–1367, 2012.

[16] D. Floreano, R. Pericet-Camara, S. Viollet, F. Ruffier, A. Brückner, R. Leitel, W. Buss, M. Menouni, F. Expert, R. Juston, *et al.*, "Miniature curved artificial compound eyes," *Proceedings of the National Academy of Sciences*, vol. 110, no. 23, pp. 9267–9272, 2013.

[17] S. Mafrica, A. Servel, and F. Ruffier, "Minimalistic optic flow sensors applied to indoor and outdoor visual guidance and odometry on a car-like robot," *Bioinspiration & biomimetics*, vol. 11, no. 6, p. 066007, 2016.

[18] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys, "An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 1736–1741, IEEE, 2013.

[19] H. W. Ho, G. C. de Croon, and Q. C. Chu, "Distance and velocity estimation using optical flow from a monocular camera," *International Journal of Micro Air Vehicles*, p. 1756829317695566, 2017.

[20] A. J. Rutkowski, M. M. Miller, R. D. Quinn, and M. A. Willis, "Egomotion estimation with optic flow and air velocity sensors," *Biological cybernetics*, vol. 104, no. 6, pp. 351–367, 2011.

[21] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 957–964, IEEE, 2012.

[22] V. Grabe, H. H. Bülthoff, D. Scaramuzza, and P. R. Giordano, "Nonlinear ego-motion estimation from optical flow for online control of a quadrotor uav," *The International Journal of Robotics Research*, vol. 34, no. 8, pp. 1114–1135, 2015.

[23] V. Grabe, H. H. Bulthoff, and P. R. Giordano, "A comparison of scale estimation schemes for a quadrotor UAV based on optical flow and imu measurements," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 5193–5200, IEEE, 2013.

[24] F. van Breugel, K. Morgansen, and M. H. Dickinson, "Monocular distance estimation from optic flow during active landing maneuvers," *Bioinspiration & biomimetics*, vol. 9, no. 2, p. 025002, 2014.

[25] G. C. de Croon, "Monocular distance estimation with optical flow maneuvers and efference copies: a stability-based strategy," *Bioinspiration & biomimetics*, vol. 11, no. 1, p. 016004, 2016.

[26] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pp. 225–234, IEEE, 2007.

[27] A. Martinelli, "Vision and IMU data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 44–60, 2012.

[28] M. Faessler, F. Fontana, C. Forster, and D. Scaramuzza, "Automatic re-initialization and failure recovery for aggressive flight with a monocular vision-based quadrotor," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 1722–1729, IEEE, 2015.

[29] J.-Y. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel Corporation*, vol. 5, no. 1-10, p. 4, 2001.

[30] B. K. Horn, Y. Fang, and I. Masaki, "Hierarchical framework for direct gradient-based time-to-contact estimation," in *Intelligent Vehicles Symposium, 2009 IEEE*, pp. 1394–1400, IEEE, 2009.

[31] H. Zhang and J. Zhao, "Bio-inspired vision based robot control using featureless estimations of time-to-contact," *Bioinspiration & Biomimetics*, 2016.

[32] A. Bruhn, J. Weickert, and C. Schnörr, "Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods," *International Journal of Computer Vision*, vol. 61, no. 3, pp. 211–231, 2005.

[33] H. C. Longuet-Higgins and K. Prazdny, "The interpretation of a moving retinal image," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 208, no. 1173, pp. 385–397, 1980.

[34] B. D. Lucas, T. Kanade, *et al.*, "An iterative image registration technique with an application to stereo vision," 1981.

[35] S. O. Madgwick, A. J. Harrison, and R. Vaidyanathan, "Estimation of imu and marg orientation using a gradient descent algorithm," in *Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on*, pp. 1–7, IEEE, 2011.

[36] K. F. Riley, M. P. Hobson, and S. J. Bence, *Mathematical methods for physics and engineering: a comprehensive guide*. Cambridge university press, 2006.

[37] D. A. Brian and J. B. Moore, "Optimal filtering," *Dower, Jan*, 2005.

[38] L. Weiss, "Controllability, realization and stability of discrete-time systems," *SIAM Journal on Control*, vol. 10, no. 2, pp. 230–251, 1972.

[39] A. J. Krener and K. Ide, "Measures of unobservability," in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pp. 6401–6406, IEEE, 2009.

[40] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Computer Vision–ECCV 2006*, pp. 430–443, 2006.

[41] M. T. Alkowatly, V. M. Becerra, and W. Holderbaum, "Bioinspired autonomous visual vertical control of a quadrotor unmanned aerial vehicle," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 2, pp. 249–262, 2014.

[42] W. R. Roderick, M. R. Cutkosky, and D. Lentink, "Touchdown to take-off: at the interface of flight and surface locomotion," *Interface Focus*, vol. 7, no. 1, p. 20160094, 2017.

[43] P. Chirarattananon, K. Y. Ma, and R. J. Wood, "Perching with a robotic insect using adaptive tracking control and iterative learning control," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1185–1206, 2016.