This paper has been accepted for publication at the
IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, 2021. ©IEEE 2021

Why fly blind? Event-based visual guidance for ornithopter robot flight

A. Gómez Eguíluz, J.P. Rodríguez-Gómez, R. Tapia, F.J. Maldonado,
J.Á. Acosta, J.R. Martínez-de Dios and A. Ollero

***Abstract*— The development of perception and control methods that allow bird-scale flapping-wing robots (a.k.a. ornithopters) to perform autonomously is an under-researched area. This paper presents a fully onboard event-based method for ornithopter robot visual guidance. The method uses event cameras to exploit their fast response and robustness against motion blur in order to feed the ornithopter control loop at high rates (100 Hz). The proposed scheme visually guides the robot using line features extracted in the event image plane and controls the flight by actuating over the horizontal and vertical tail deflections. It has been validated on board a real ornithopter robot with real-time computation in low-cost hardware. The experimental evaluation includes sets of experiments with different maneuvers indoors and outdoors.**

***Index Terms*— ornithopter, flapping-wing, event camera, line features, tracking, visual servoing, computer vision.**

I. Introduction

In the last years, flapping-wing robots have attracted significant R&D interest. They have the potential of performing forward, backward, and lateral flight, other agile maneuvers, and hovering [1]. They are robust against collisions and less dangerous than multirotors. Moreover, their flapping and gliding capabilities can be combined to perform energy-efficient flights. A number of works have explored small scale flapping-wing Micro Aerial Vehicles (MAVs), see e.g., [2], [3]. Some of them have even explored visual perception for flapping-wing MAVs [4]. We are interested in bird-scale flapping-wing robots, also called ornithopters, which have enough payload capability to install onboard sensors and embedded computers that can enable their fully autonomous operation.

Flapping-wing flight entails a number of perception challenges that differ from multirotor flight. Ornithopters suffer from mechanical vibrations and wide abrupt movements due to the flapping strokes , which highly impact onboard perception [5]. Besides, their strict payload and energy limitations severely constrain the sensors, gimbals, and computing or additional hardware that can be installed on board. Further, increasing the payload reduces the ornithopters' maneuverability. Although the vibrations and abrupt motion issues are less acute when the ornithopter flies in gliding mode than in flapping mode, their strict payload limitations preclude the installation of different sensors for each flight mode.

This work was funded by the European Research Council as part of GRIFFIN ERC Advanced Grant 2017 (Action 788247) (https://griffin-erc-advanced-grant.eu).
The authors are with the GRVC Robotics Lab Sevilla. Universidad de Sevilla, Seville 41092, Spain email: {ageguiluz, jrodriguezg, raultapia, f.j.maldonado.fernandez, jaar, jdedios, aollero}@us.es

Fig. 1: The GRIFFIN *E-Flap* ornithopter during an outdoor fully onboard guidance experiment.

Finally, the ornithopter payload entails strict limitations on the onboard processing capabilities. In fact, most of the reported ornithopter control methods, see e.g., [6]–[8], are executed offboard using measurements from very accurate external perception systems such as motion capture systems.

The proposed perception scheme is based on event cameras. Event cameras are robust to motion blur and lighting conditions; they are compact, have moderate weight, and low energy consumption. Hence, they are suitable onboard sensors to deal with the perception challenges of flapping-wing flight [5]. Besides, efficient event-based processing techniques can provide estimates at high rates, as required to cope with ornithopters' agility and maneuverability. A good number of successful event-based perception techniques have been developed, see e.g. [9]. Although they have been used on board multirotors, no event-based technique dealing with the perception issues of ornithopters has been reported.

This paper presents a fully onboard event-based scheme for guidance of ornithopter robots. It tracks intersections of line segments, which are used in a visual servoing system to compute the velocity commands to the ornithopter controller. The scheme closes the control loop fully onboard at rates of 100 Hz in low-cost lightweight hardware. It has been evaluated using the GRIFFIN *E-Flap* ornithopter [7] in both indoor and outdoor scenarios, see Figures 1 and 3. The contribution of the paper is three-fold: an event-based line tracking method that enhances the robustness and efficiency of the method presented in [10]; an efficient visual servoing method that exploits event-based vision to perform ornithopter guidance; and the experimental validation in short flapping and gliding maneuvers indoors and outdoors. To the best of the author's knowledge, this is the first method in which a bird-scaled flapping-wing robot is controlled using a fully onboard event-based perception system.

This paper is structured as follows. Section II summarizes the main works in the topics addressed in the paper. The general scheme of the event-based ornithopter guidance system and its main components are described in Sections III and IV, respectively. Section V presents the experimental validation and robustness analyses. Finally, Section VI concludes the paper and highlights the main future research steps.

II. Related Work

In recent years, the development of a wide variety of flapping-wing robots has motivated increasing interest in methods for their control and guidance. A number of methods for ornithopter flight control, obstacle avoidance, or control during maneuvering among others, have been developed [3], [4], [6]–[8]. However, most existing works used sophisticated methods involving significant computational requirements that preclude their onboard execution [3] [8], relied on measurements from external sensors to close the control loop [6] [7], or performed onboard image-based processing running at typical frame rates ($\leq$ 30 Hz) [4]. Our objective is to guide and control bird-scaled ornithopters using solely onboard sensors while processing at high frequency rates.

The perception of the environment by ornithopter robots is challenging as abrupt movements and strong mechanical vibrations are recurrent during the flight. The intrinsic nature of ornithopter flight requires the development of perception systems for both gliding and flapping-wing flight, ideally using the same sensors to keep the payload as low as possible. One of the first approaches to robotic visual perception for ornithopters was *ROSS-LAN*, a simulation scheme to obtain synthetic data of a number of sensors during landing and perching maneuvers [11]. The work in [12] developed a vision stabilizing system to address the pitch and roll fluctuations during each flapping period. The required payload for installing the hardware for their proposed vision system was $<$100 g, which could be carried within the 150 g payload of their robot. Although their system provides a valid solution to some of the problems that arise during flapping-wing flight, increasing the payload often entails that the maneuverability and autonomy are reduced [7]. The work in [5] studied the potential issues and opportunities of using LiDAR, conventional, and event-based vision sensing on board ornithopter robots. They concluded that event-based vision provides a promising solution to many of the perception challenges that arise during flapping-wing flight.

The advent of event cameras has recently attracted significant research interest in the robotics and computer vision communities [9]. A number of works have explored the advantages of event cameras on aerial robots. A method for detection and tracking of moving objects onboard a Micro Aerial Vehicle (MAV) was presented in [13]. A model of the affine transformation between two consecutive *event images* was used to compensate for the global motion of a MAV and, the resulting events were assumed to represent the moving objects. In [14], an autonomous MAV landing approach based on the optical flow of event frames obtained from a downwards-pointing DVS sensor was presented. The authors compared to other landing approaches based on frame-based cameras showing that their approach was the most accurate at high speed. The work in [15] proposed a high-speed dodging system for UAVs. A Deep Learning solution used *event images* to detect independent moving objects, estimate their 3D motion, and avoid collisions. Another dodging system for UAVs was presented in [16]. The authors relied on the spatio-temporal continuity of events to detect and track moving objects and proposed a method based on potential fields to execute the evasive maneuver. Recently, *event images* have been used in [17] to estimate the position and yaw orientation of a quadrotor using Visual-Inertial Odometry (VIO) and closing the loop for autonomous flight subject to failures.

Although the output of event cameras are asynchronous event streams, all the above techniques group the temporally-close received events in frames called *event images*. Hence, they do not fully exploit the sequential and asynchronous nature of event cameras and, in fact, some of them [15] include motion blur cancellation mechanisms. Various asynchronous *event-by-event* methods have been proposed for feature detection, [18] feature tracking [19] [10], clustering [20], pose tracking [21], and VIO [22]. Although these methods provide valid solutions, few of them have considered computational constraints like those on board aerial robots.

The work presented in [21] performs onboard tracking of a drone's 6-DoF pose during high-speed maneuvers by looking at a previously known planar shape on a wall. More recently, a bio-inspired visual servoing method was presented in [10] mimicking the approach followed by pigeons while perching and relying on the time-to-contact to guide a multirotor UAV in vertical descent maneuvers. The work in [23] implements a closed-loop control of a dualcopter. Using an event camera, their method was capable of estimating the robot state on board and enabling attitude tracking at speeds of 1600 deg/s.

Although all these works present relevant contributions to event-based vision for robotics, none of them have been designed for or tested in ornithopter robots. The *GRIFFIN perception dataset* [24] includes data collected from a conventional camera, event camera, and two Inertial Measurement Units (IMU) on board an ornithopter. Despite being a first approximation, the use of onboard sensing for close loop control in ornithopter robots is still an under-researched area. In fact, to the author's knowledge, there are no previous works in which a bird-scale flapping-wing robot is controlled using a fully onboard perception system.

III. General Description

Autonomous maneuver execution of ornithopter robots using a fully onboard perception and control system is significantly different from how the same problem is approached in multirotor UAVs, and involves dealing with relevant issues. In ornithopters, the lift and thrust are generated by flapping strokes. They are strongly underactuated platforms, their 6-DoF motion should be controlled by a few number of actuations. Flapping-wing flight suffers from mechanical vibrations and wide abrupt movements that impose limitations in

This paper has been accepted for publication at the
IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, 2021. ©IEEE 2021

the perception systems [5]. Ornithopters are also affected by aerodynamic disturbances, which impact the control methods. Moreover, they have strict payload limitations, which constrain the installation of onboard sensors and hardware.

The proposed scheme is based on event cameras, which are neuromorphic sensors that capture the visual information in the form of events representing changes of illumination in the scene. Events are triggered asynchronously with high temporal resolution (μs), hence event-based processing can provide estimates at very high frequencies. Moreover, they are robust to changes in lighting conditions due to their high dynamic range ($\sim$120 dB), and do not suffer from motion blur. Finally, they are compact, have moderate weight, and low energy consumption.

The adopted event-based processing scheme is based on an efficient Image Based Visual Servoing (IBVS) method in which the goal position is defined w.r.t. a reference pattern. The reference pattern is assumed to be defined by the intersections of a set of straight-line segments. Lines contain richer structure than punctual features such as corners, and can be more robustly extracted than them. Besides, a wide variety of objects originate lines in the event camera, some of which, such as landing pads, are actually used for aerial robot maneuvers. Using line intersections provides the accuracy of feature points with the robustness of line extraction. The ornithopter platform actuates over the tail pitch and direction of the rudder while flapping is at a constant rate. Thus, translation and rotation are coupled. The method assumes that the maneuver meets the kinematic and dynamic restrictions of the platform while keeping the reference pattern in the field of view (FoV) of the camera. The initial maneuver position is assumed to allow the reference pattern to be detected in a reliable manner. Under these assumptions, the proposed IBVS approach can be simplified to provide translational reference commands as the constrained control actions do not entail orientation changes that require significant robot attitude corrections. Finally, the event processing scheme is endowed with ASAP [25], which synchronizes event packaging such that events are processed as soon as possible while avoiding overflow. ASAP enables executing the proposed event-based method at a guaranteed frequency of 100 Hz.

The adopted controller meets both robustness and simplicity requirements, and actuates only on the horizontal and vertical tail deflections. The efficient controller, based on well-established high-gain nonlinear control theory [26], is *explicit* thus enabling high execution frequency on the onboard hardware where many other processes are running simultaneously. A feedback frequency of 100 Hz allows a continuous-time design methodology, and hence to assess its robustness using Lyapunov-like stability theory, see Section IV-C.

IV. Methods

A. Event-based line detection and tracking

This section presents a method to track the lines of a reference pattern along the full guidance trajectory. Based on the work we proposed in [10], this new method was enhanced to improve line tracking robustness and reduce burden. It includes two main modules. First, a spatio-temporal consistency filter is applied to remove events caused by the sensor's noise and events triggered from objects with low spatio-temporal consistency in the event stream. Next, an Extended Kalman Filter (EKF) is used to track lines fusing fast-response (but noisy) *event-by-event* line estimates together with robust (but slower) line measurements obtained from event-image processing.

An event binary filtering mask $\mathbf{\Omega}$ is computed by accumulating the last received events in a binary *event image*, and applying erode and dilate morphological operations to delete noise and enhance regions with a high number of event neighbors. $\mathbf{\Omega}$ is updated every 10 ms, it has the same size as *event images*, and contains a spatio-temporal representation of the scene structure computed from the events triggered during the last 40 ms. An input event at coordinates (u, v) is considered valid if it has spatio-temporal consistency with $\mathbf{\Omega}$, i.e. if $\mathbf{\Omega}(u, v) = 1$. The rest are filtered out. Figure 2 shows the result of the adopted event filter in one experiment.

The line extraction and tracking methods use the polar representation of lines: $\rho = u\cos\theta + v\sin\theta$, with each line defined by $l(\theta, \rho)$ in the Hough space. First, the lines to be tracked by the EKF, represented as $L_T = (l_0, ..., l_n)$, are initialized with the n lines that define the reference pattern. The EKF keeps track of the lines in L_T by integrating in the prediction stage the line estimates resulting from *event-by-event* processing, and in the update stage, the estimates resulting from event-image processing.

The events received by the EKF are divided into two sets with 50% probability: e^P and e^U, which are used in the EKF prediction and update stages, respectively. In the EKF update, an *event image* $\mathbf{S}$ is updated every 10 ms with the events e^U received during last 20 ms to render a suitable representation of L_T in $\mathbf{S}$. Line candidates are extracted from $\mathbf{S}$ using the Hough transform adapted –to reduce computational cost– with a clustering phase to avoid redundant candidate lines in the Hough space. The set of extracted candidate lines L_C are used in the EKF update. First, candidate lines are evaluated to find possible associations with reference lines in L_T. If the distance in the Hough space between a candidate line and tracked line l_i in L_T is lower than a threshold η_T, the candidate line is associated to l_i. If only one candidate line is associated to l_i, l_i (and also its covariance) is updated. In case several line candidates are associated to l_i, the line candidate

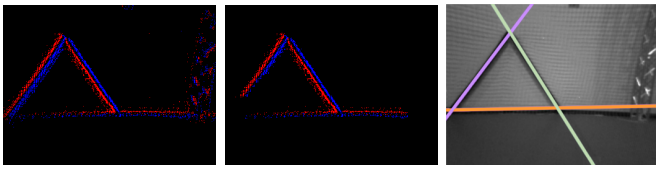

Fig. 2: Results from the event line tracker during a guidance experiment: left) input events; center) events after applying the spatio-temporal consistency filter; and right) resulting tracked lines. For better visualization left) and center) are shown as *event images* accumulated during 20 ms, and right), overlaid on a visual frame image from the DAVIS 346 APS.

This paper has been accepted for publication at the
IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, 2021. ©IEEE 2021

closer to l_i is used for update, and the rest is discarded.

Conversely, events in e^P are used in the EKF prediction. Each event is evaluated to determine if it can be associated to a line in L_T or not. The association consists of evaluating if the event lies in a region in the Hough space close to any of the lines in L_T. If the event is associated with just one line, it updates the line and its covariance matrix. If the event is associated to more than one line or is not associated to any line, it is considered as noise and discarded. If a tracked line has not been updated in the last 100 ms, it is deleted and substituted by a new line in the Hough neighborhood of the deleted line, selected using minimum distance criteria.

The proposed line tracker discards events caused by noise, increasing robustness and saving computational effort. As shown in Section IV-A, it obtains burden reductions of $\sim$40% over the method in [10], reduces line prediction error in $\sim$5%, and increases the tracked line lifetimes in $\sim$10%.

B. Visual servoing using event-based vision

This section presents a visual servoing method to guide the ornithopter towards the goal pose using the difference in the image plane between the current features and their goal positions. The adopted features are intersections of lines extracted as in Section IV-A, and correspond to coplanar 3D points of a reference target. As with many other IBVS techniques, our method assumes known the geometry of the target, and the ornithopter goal pose in the reference frame attached to the plane defined by the 3D points. Using a calibrated camera, the goal position of features in the image plane are computed projecting the 3D points as if the robot were at the goal pose before the maneuver starts.

First, the intersections of the tracked lines are computed. For each tracked line l_i, two points $(u_{i,1}, v_{i,1})$ and $(u_{i,2}, v_{i,2})$ laying in l_i are obtained by choosing $u_{i,1}$ and $u_{i,2}$ and using the Hough space equation (see Section IV-A) to obtain $v_{i,1}$ and $v_{i,2}$. The intersection point $\mathbf{p}$ of two non-parallel lines l_i and l_j is computed as follows:

$$\mathbf{p} = \begin{bmatrix} \frac{\begin{vmatrix} \mathbf{L}_A & \mathbf{X}_A \\ \mathbf{L}_B & \mathbf{X}_B \end{vmatrix}}{\begin{vmatrix} \mathbf{X}_A & \mathbf{Y}_A \\ \mathbf{X}_B & \mathbf{Y}_B \end{vmatrix}}, & \frac{\begin{vmatrix} \mathbf{L}_A & \mathbf{X}_A \\ \mathbf{L}_B & \mathbf{X}_B \end{vmatrix}}{\begin{vmatrix} \mathbf{X}_A & \mathbf{Y}_A \\ \mathbf{X}_B & \mathbf{Y}_B \end{vmatrix}} \end{bmatrix}, \quad \mathbf{L}_A = \begin{vmatrix} u_{i,1} & v_{i,1} \\ u_{i,2} & v_{i,2} \end{vmatrix}, \quad \mathbf{L}_B = \begin{vmatrix} u_{j,1} & v_{j,1} \\ u_{j,2} & v_{j,2} \end{vmatrix}, \tag{1}$$
$$\mathbf{X}_A = \begin{vmatrix} u_{i,1} & 1 \\ u_{i,2} & 1 \end{vmatrix}, \quad \mathbf{Y}_A = \begin{vmatrix} v_{i,1} & 1 \\ v_{i,2} & 1 \end{vmatrix}, \quad \mathbf{X}_B = \begin{vmatrix} u_{j,1} & 1 \\ u_{j,2} & 1 \end{vmatrix}, \quad \mathbf{Y}_B = \begin{vmatrix} v_{j,1} & 1 \\ v_{j,2} & 1 \end{vmatrix},$$

where for a matrix $\mathbf{A}$, $|\mathbf{A}|$ denotes its determinant.

Next, the ornithopter velocity commands are computed using IBVS. Let $\mathbf{p}_i(t)$ be the coordinates u and v of feature i detected at time t. Let $\mathbf{p}_i^*$ be the goal position of feature i. We can define $e(t)$ as an n x 2 matrix with the difference between $\mathbf{p}_i(t)$ and $\mathbf{p}_i^*$. The camera velocity error at time t can be computed as, [27]:

$$\nu(t) = -\mathbf{K}\mathbf{J}^{\dagger} e(t), \tag{2}$$

where $\mathbf{K}$ is a positive definite diagonal weighting matrix, and $\mathbf{J}^{\dagger}$ is the pseudoinverse of $\mathbf{J}$, the interaction matrix that describes the variation of the feature position as a function of camera velocity. The resulting camera velocity error $\nu(t)$ is sent to the ornithopter controller (see Section IV-C).

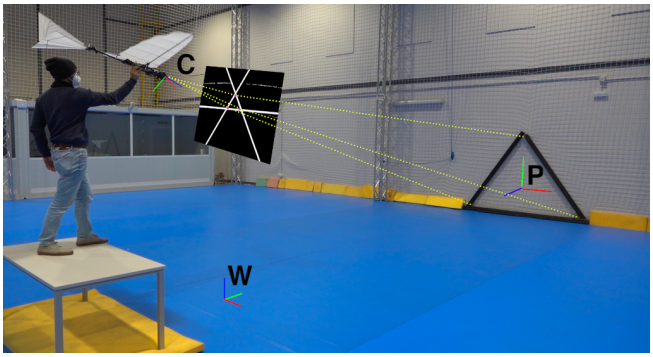

Fig. 3: Diagram of the coordinate frames used by the proposed event-based visual servoing method. C is the camera frame, P is the frame attached to the pattern's plane, and W is the world frame used by the Optitrack system.

$\mathbf{J}$ is computed as follows. The kinematics of image feature $\mathbf{p}$ can be expressed as $\dot{\mathbf{p}} = \mathbf{J}_{\mathbf{p}}\nu$, where ν is the camera (linear and angular) velocity vector and $\mathbf{J}_{\mathbf{p}}$ is the Jacobian that describes the variation of the position of $\mathbf{p}$ as a function of the camera velocity, and is computed as:

$$\mathbf{J}_{\mathbf{p}} = \begin{bmatrix} -\frac{\lambda}{\rho_u d} & 0 & \frac{u}{d} & \frac{\rho_u uv}{\lambda} & -\frac{\lambda^2+\rho_u^2 u^2}{\rho_u \lambda} & v \\ 0 & -\frac{\lambda}{\rho_v d} & \frac{v}{d} & \frac{\lambda^2+\rho_v^2 v^2}{\rho_v \lambda} & -\frac{\rho_v uv}{\lambda} & -u \end{bmatrix}, \tag{3}$$

where λ is the camera focal distance, ρ_u and ρ_v are the pixel size of the camera (in μm), and d is the distance between the camera and the position of the 3D point corresponding to $\mathbf{p}$ expressed in axis Z of the camera reference frame C.

It is worth considering that, despite the line feature motion is subject to both translation and rotation, ornithopters are underactuated systems. For instance, the robot must tilt to control the altitude, and the roll and yaw rotations are coupled to control the lateral motion. A common practice when using IBVS in multirotor UAVs is to compensate for the rotational motion by projecting the feature into a virtual frame, e.g. [10] [28]. However, the kinematics of the ornithopter precludes the use of such rotation compensation without significantly increasing the risk of missing the feature tracks, e.g. due to the pattern leaving the camera Field of View (FoV). Therefore, a compromise solution should guide the ornithopter in translation while keeping the features within the camera FoV. This is taken into account when computing the Jacobian $\mathbf{J}_{\mathbf{p}_i}$ by setting its three last columns (those corresponding to the angular velocities) to zero. This is consistent with the adopted ornithopter controller, which actuates over the tail pitch and direction of the rudder, and saves computational burden enabling faster control loop closing. Finally, we are dealing with n features, $\mathbf{p}_i$ with $i \in [1, n]$. Hence, the interaction matrix $\mathbf{J}$ is built through row-wise concatenation of the Jacobians for all the n features.

Computing $\mathbf{J}_{\mathbf{p}_i}$ requires having d_i, the distance between the camera and the 3D point corresponding to $\mathbf{p}_i$, expressed in axis Z of the camera frame. IBVS is well known to be quite robust to errors in the estimation of d_i. For efficiency, our method computes the distance between the robot and plane Π, which contains the n 3D points, and uses that distance for all the features. Figure 3 shows a diagram

of the reference frames adopted. Let P be the reference frame attached to plane Π with origin at the centroid of the n 3D points. The camera pose w.r.t. P can be obtained using the Direct Linear Transformation (DLT) algorithm [29]. The transformation between a point $\begin{bmatrix} p^x & p^y \end{bmatrix}^T$ in Π expressed in coordinates in P and its projection $\begin{bmatrix} u & v \end{bmatrix}^T$ on the camera plane can be expressed as $\omega \begin{bmatrix} u & v & 1 \end{bmatrix}^T = \mathbf{H} \begin{bmatrix} p^x & p^y & 1 \end{bmatrix}^T$, where ω is a scaling factor and $\mathbf{H}$ is the homography matrix between Π and the image plane. From the correspondence between 3D point $\mathbf{p}_i$ in Π and its corresponding point in the image plane, $\mathbf{M_p}$ is built:

$$\mathbf{M_p} = \begin{bmatrix} -p^x & -p^y & -1 & 0 & 0 & 0 & p^x u & p^y u & u \\ 0 & 0 & 0 & -p^x & -p^y & -1 & p^x v & p^y v & v \end{bmatrix} \quad (4)$$

Let $\mathbf{M}$ be a $2n \times 9$ matrix built by row-wise concatenating $\mathbf{M}_{\mathbf{p}_i}$ for each of the n correspondences. Computing the Singular Value Decomposition (SVD) $\mathbf{M} = \mathbf{U\Sigma V}^T$ and assuming $n \geq 4$ –e.g., three line intersections and the centroid of the intersections, $\mathbf{H}$ can be obtained from the entries in the last column of $\mathbf{V}$. Using the camera matrix $\mathbf{A}$, $\mathbf{H}$ can be decomposed into the rotation matrix and the translation vector between P and C. Finally, d is obtained as the module of the vector resulting from projecting the translation vector on axis Z of the camera frame C.

The computation of $\mathbf{J}^\dagger$ requires at least three points. This is a well known limitation in IBVS and, in fact, choosing more than three points is a common practice to avoid singularities of $\mathbf{J}$, [27]. Besides, the computation of d also requires at least three features –using their centroid as forth point. Hence, our method requires ≥ 3 non-overlapping lines, which create ≥ 3 intersections, sufficient to compute d and $\mathbf{J}^\dagger$.

C. Ornithopter control

The ornithopter controller uses $\nu(t)$ from (2) provided by IBVS to compute the control actions. The high maneuverability and aggressive motion of ornithopter robots require control frequencies >100 Hz. Unlike works such as [6], [7], or [8], our method closes the control loop using solely onboard sensors and processing, without any external measurements or sensors. Controlling ornithopters using onboard perception with strict payload and energy limitations have high uncertainty levels. Robustness to minimize the influence of uncertainty and computational efficiency are main requirements in the adopted controller. One control method that accomplishes both requirements is the high-gain feedback controller [26]. Besides, the proposed onboard perception system provides $\nu(t)$ at guaranteed frequencies of 100 Hz enabling the use of a continuous-time design methodologies. Thus, the proposed controller is derived in continuous time, and robust stability results are provided through Lyapunov-like theory. Let $\mathbf{x} \in \mathbb{R}^n$ denote the full state vector of the ornithopter, and $\mathbf{v} = [\mathrm{v}_y, \mathrm{v}_y]^T \in \mathbb{R}^2$ the velocity of the ornithopter flight obtained from the (x, y) linear velocity components of $\nu(t)$, provided by the onboard perception algorithms computed as in Section IV-B. Thus, the velocity dynamics can be formulated from the flight equations of motion in the camera reference frame [30] as[1]

$$\dot{\mathbf{v}} = f(\mathbf{x}) + g(\mathbf{x}) \circ \mathbf{u} + \mathbf{\Delta}(\mathbf{x}, t), \quad (5)$$

where f and g are *smooth* vector functions of the appropriate dimension, $\mathbf{u} \in \mathbb{R}^2$ is the control input vector, $\mathbf{\Delta} \in \mathbb{R}^2$ is an additive disturbance including e.g. the perception uncertainty among others. The control actions are the horizontal (δ_e) and vertical (δ_r) tail deflections so that $\mathbf{u} = [\delta_e, \delta_r]^T$.

Consider the ornithopter velocity dynamics (5) in the flight envelope, with $\mathbf{v}$ the only available measure, i.e. the state vector $\mathbf{x}$ is not available for feedback since no external sensors are used. The control objective is to design $\mathbf{u}(\mathbf{v})$ so that (5) is *practically stable* and hence, there exist positive constants $\mathbf{B}$ and T such that $\|\mathbf{v}(t)\| \leq \mathbf{B}$ for some $t \geq T$.

From the statement above, it is evident the need of a robust controller, because of the high uncertainty and the lack of measures. Roughly speaking the controller is 'blind' to the remaining states. Additionally, flying in the flight envelope ensures that: i) the state $\mathbf{x}$ is bounded, thus simplifying the controller design, and; ii) $g(\mathbf{x})$ is away from zero with known signs elementwise. The latter allows us to assume without any loss of generality that $g(\mathbf{x}) > 0$, to ease the controller derivation. Thus, consider the positive definite and radially unbounded Lyapunov function $V = (\mathbf{v} \cdot \mathbf{v})/2$. Its derivative along the trajectories of (5) becomes

$$\dot{V} = \mathbf{v} \cdot (f + \mathbf{\Delta}) + \mathbf{v} \cdot (g \circ \mathbf{u}) \leq |\mathbf{v}| \cdot \eta + \mathbf{v} \cdot (g \circ \mathbf{u}), \quad (6)$$

where from boundedness i) above, the following upper bound follows $\|f + \mathbf{\Delta}\| < \eta$, for some $\eta > 0$. It is not difficult to see from (6) that the high-gain control structure as proposed in [26] is suitable, which can be defined as

$$\mathbf{u} = -\boldsymbol{\beta}(\mathbf{v}) \circ \mathbf{v}, \quad (7)$$

for any positive vector function $\boldsymbol{\beta}(\mathbf{v})$ with $\boldsymbol{\beta}(\mathbf{0}) > 0$, such that $\|g \circ \boldsymbol{\beta}(\mathbf{v})\| \geq \eta$. To meet robustness and simplicity requirements, we define $\boldsymbol{\beta}(\mathbf{v}) = \boldsymbol{\beta_0} + \boldsymbol{\beta_1}|\mathbf{v}|$.

Certainly, this controller behaves as desired, proportionally around $\mathbf{v} \approx \mathbf{0}$ and reacting aggressively to leave that neighborhood. Finally, recalling the fact ii) above on g, it is straightforward to see that there always exists $\boldsymbol{\beta_0}$ satisfying $\|g \circ \boldsymbol{\beta}(\mathbf{v})\| \geq \eta$ in a neighborhood of $\mathbf{v}$, and $\boldsymbol{\beta_1}$ such that outside of that neighborhood $\dot{V} \leq 0$. Moreover, an estimate of the ultimate bound can be computed by using Young's inequality in (6) with (7) as $\mathbf{B} = \eta^2/(\beta_0^2 \bar{g})$ with $\bar{g} = \max_{\mathbf{x}}\{g(\mathbf{x})\}$. Therefore, under the aforementioned conditions, we can conclude ultimate boundedness of trajectories which means that the control objective is achieved.

V. Experimental results

The proposed scheme was validated in the *E-Flap* ornithopter robot [7], a custom design ornithopter developed by the GRVC Robotics Lab. It has an empty weight of 510 g, a total length of 95 cm, a maximal wing span of 1.5 m, and a maximum payload of 520 g (with reduced maneuverability and flight time). *E-Flap* was modified to equip a DAVIS 346

[1]Notation: $\circ$ denotes the Hadamard product and $\cdot$ the dot product. For a vector $\mathbf{z}$, $|\mathbf{z}|$ denotes the element-wise absolute value and $\|\mathbf{z}\|$ its norm.

This paper has been accepted for publication at the
IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, 2021. ©IEEE 2021

(a) *Outdoors*. (b) *Indoors2*. (c) *Indoor3*.

Fig. 4: Sequences of the experimental evaluation of three flapping-wing flight maneuvers: *Outdoors*, *Indoors2*, and *Indoors3*.

event camera, and its onboard computer was replaced by a Khadas VIM3 for onboard perception and control processing. The platform used the same power supply for both actuation and perception. Similarly to the ornithopter used in [24], the onboard computer and sensors were carefully installed to keep balance and guarantee the stability and maneuverability of the platform with an additional payload of $\sim$180 g. The DAVIS346 embeds a dynamic vision sensor (DVS) that outputs timestamped and polarized events, and an active pixel sensor (APS) coincident with the DVS that outputs grayscale images at 40 Hz. The low-weight Khadas VIM3 board mounted Ubuntu 18.04. The event-based processing and the controller were programmed in C++ using *ROS Melodic*.

The underactuated nature of the ornithopter and the functionality requirements of IBVS were considered for the validation of the proposed method. First, the *E-Flap* kinematic constraints described in [7] along with the onboard perception restrictions influence the duration of the maneuver. From the dynamic point of view, *E-Flap* typically requires a flying speed of 4 m/s while flapping to maintain a constant height. To correctly detect the lines, the maximum distance between the onboard DAVIS346 and the reference pattern was 10 m. Under these considerations, the guidance maneuvers in our experiments described trajectories of $\sim$2 sec. Second, keeping the reference pattern in the camera FoV along the maneuver is hindered by the robot's attitude changes due to flapping. Taking into account the ornithopter kinematics, to keep the reference pattern within the camera FoV, we limited the tail actuation to horizontal deflection $\delta_e \in [-30, 10]$ deg and lateral deflection $\delta_r \in [-20, 20]$ deg.

The event-based ornithopter guidance scheme was validated in sets of experiments performing different maneuvers in two scenarios: a testbed and an outdoor scenario. The testbed was a $15 \times 21 \times 8$ m room designed for testing ornithopters equipped with 24 *OptiTrack Prime*^{x}13 cameras that provided millimeter accuracy robot pose estimations, used only as ground truth for evaluation. The outdoor scenario was chosen to validate the robustness of the scheme to the uncertainties arisen in open uncontrolled spaces.

A total of 20 flights were performed covering different gliding and flapping maneuvers indoors and outdoors. Four different experiments were evaluated: *Indoors1*, gliding descending maneuvers indoors; *Indoors2*, flapping during smooth descending maneuvers indoors; *Indoors3*, flapping during horizontal flight maneuvers indoors; and *Outdoors*, flapping during smooth descending maneuvers outdoors. Figure 4 shows three sequences of experiments *Outdoors*, *Indoors2*, and *Indoors3*. For brevity, a sequence for *Indoors1* experiments is not shown as it is similar to *Indoors2*. In all experiments, the goal position was chosen near the reference pattern (a triangle), keeping it centered in the image plane, and the ornithopter being parallel to the ground. All the experiments were conducted using the same algorithm parameters.

A. Feature detection and tracking evaluation

First, we evaluated the robustness of the proposed line tracking method to the vibrations and changes in lighting conditions that can be found in ornithopter flights. Figure 5-top-left shows the module of the linear acceleration registered by a VectorNav VN-200 IMU onboard the ornithopter during a guidance experiment of type *Indoors2*. The evolution of the three tracked lines in the Hough space along the experiment is shown in Figure 5 (top-right and bottom). Despite the high vibration level, the line tracker provided smooth line estimates. Additionally, due to the wide dynamic range of event cameras, the adopted method succeeded in tracking lines with high robustness to lighting conditions. As an example, Figure 6 shows its operation in two scenarios with very different lighting conditions. The robustness to vibrations and lighting changes was observed in all the experiments performed. We also compared the robustness of the proposed method against the method described in [10]. The proposed method obtained tracked lines lifetimes $\sim$10% longer and line prediction errors $\sim$5% lower, denoting

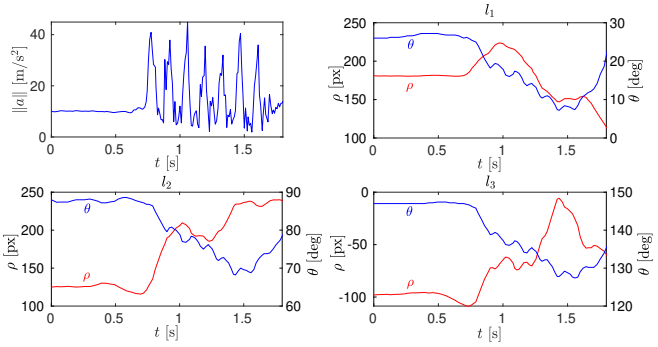

Fig. 5: Top-left) Module of the linear acceleration registered onboard the ornithopter during a guidance experiment. Top-right and bottom) Temporal evolution of the Hough (θ, ρ) components of the three tracked lines along the experiment. The ornithopter was launched at $t \approx 0.7$ s.

This paper has been accepted for publication at the
IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, 2021. ©IEEE 2021

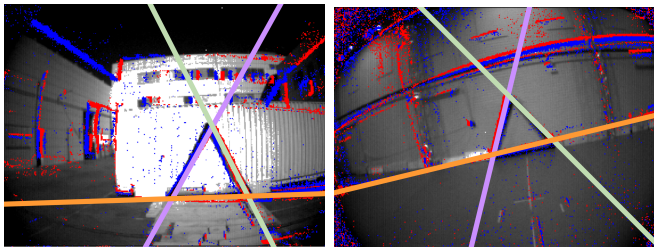

Fig. 6: Line tracking in outdoors under high intensity sunlight (left) and indoor illumination conditions (right). For visualization, the lines are drawn over a frame from the APS along with the events from the DVS.

stronger robustness. The computational efficiency was also compared. Both methods were executed on the *Khadas VIM3* without other additional processes running in parallel, and ASAP was configured to provide event packages at 500 Hz. The proposed method obtained burden reductions of ∼40%, providing average execution rates of ∼450 Hz, while the method in [10] provided rates of ∼350 Hz. Thus, the proposed line tracker absorbs the main sources of perturbations in flapping-wing flight and provides high rates of accurate, robust, and smooth line estimates suitable for ornithopter guidance.

B. Autonomous maneuvers

Figure 7-top shows the values of the camera velocity error $\nu(t)$ resulting of the IBVS method (used as input reference to the controller) along the maneuver shown in Figure 5. The figure is split in two stages in purple and yellow to differentiate between the periods before launching and during the maneuver. The camera velocity errors shown in Figure 7-top approached zero values during the maneuver, which confirms that the robot is approaching the goal position while keeping the line tracks. As the computation of $\nu(t)$ is dependent on the feature error and the estimated distance to the pattern, the reference commands were greater at the beginning of the maneuver than at the end –hence, the control actions too. The control actions δ_e and δ_r were obtained from (7), see Figure 7-center. Recall from Section IV-C that we choose to control the vertical and lateral components of the error, v_y and v_x respectively. The controller gains $\boldsymbol{\beta}_0$ and $\boldsymbol{\beta}_1$ from (7) were tuned experimentally. The followed criteria was to achieve a soft response from the controller when the error $\mathbf{v}$ is small and aggressive (fast) response for $\mathbf{v}$ large, property achieved thanks to the nonlinear nature of the controller (see Section IV-C) for details. Notice that the deflection δ_e saturated during the transient to keep ornithopter in the flight envelope.

Figure 7-top shows the smoothed input errors $\mathbf{v}$ provided by the perception system. It can be seen than during the initial stages of the flight, the magnitude of v_y was large compared to the magnitude of v_x. This implies that the initial deviation in the longitudinal dimension is larger and requires a more aggressive control action to converge. After an initial transient, both of the controlled states, v_x and v_y, were confined in a region around the origin, as predicted by design. An estimate of this region can be computed as

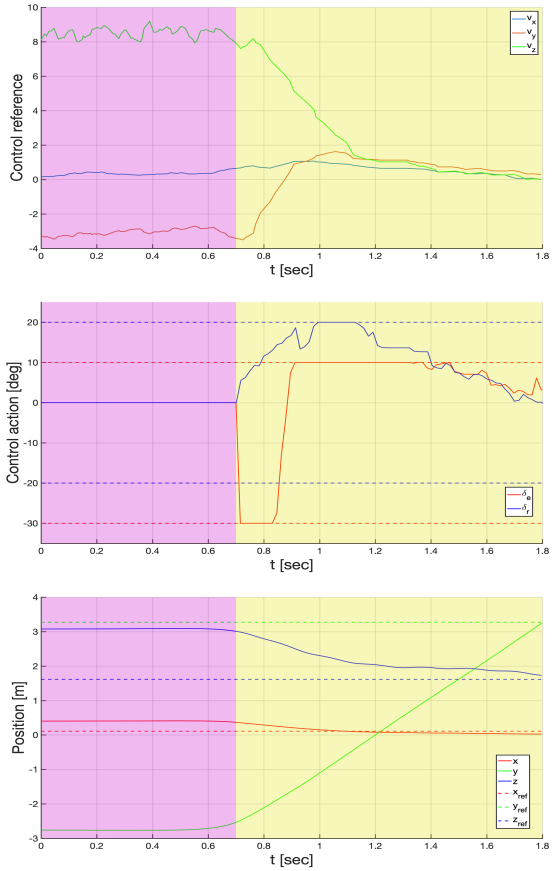

Fig. 7: Evolution of $\nu(t)$ (top), $\delta_e(t)$ and $\delta_r(t)$ (center), and ornithopter 3D positions (bottom) along the maneuver in Figure 5. Purple and yellow colors differentiate between before and after launching. The ornithopter 3D positions were captured by the Optitrack system in the frame W.

described in Section IV-C. Figure 7-bottom shows the 3D positions of the ornithopter in the reference frame W along with the maneuver. The 3D position was obtained using *OptiTrack*. The goal position is represented with dashed lines. The ornithopter controlled its trajectory to perform a smooth maneuver and reached the goal position when the longitudinal position (i.e., y) reached the reference. At the end of the maneuver the ornithopter had a 3D position error w.r.t. the goal position of 0.207 m.

Table I shows the average results of different error metrics in the conducted experiments. The table shows the root mean square error (RMSE) and the normalized root mean square error (NRMSE) normalized by $max(\mathbf{v}) - min(\mathbf{v})$ to measure control performance in both dimensions. The RMSE measures the performance in terms of the magnitude of the final error, and the NRMSE allows to measure how small the final values of $\mathbf{v}$ are relative to the range of values of $\mathbf{v}$ during flight. Only the last 20 samples were taken to compute the errors. The results show an average NRMSEy of 8%, NRMSEx of 21%, RMSEy of 0.64, and an RMSEx of 0.31 across all tests (indoors and outdoors). The obtained errors in both dimensions are small in magnitude, but the range of values found in the tests for v_y is larger than

	Indoors1	*Indoors2*	*Indoors3*	*Outdoors*
Final error (m)	0.465	0.221	0.409	–
Controller NRMSEx	0.2482	0.1745	0.3810	0.0320
Controller RMSEx	0.3356	0.4192	0.4291	0.0227
Controller NRMSEy	0.1384	0.0300	0.1209	0.0331
Controller RMSEy	1.1665	0.2648	0.8214	0.3225

TABLE I: Average performance analysis of the proposed method for the experiments.

for v_x. This is a direct consequence of the dynamics of the robot, the constraints of the tests, and the restrictions imposed in δ_e and δ_r. Because of the above restrictions, the robot was flying with reduced capabilities. The lateral-directional dynamics of the robot are less maneuverable than the longitudinal ones. It can be also noticed that flapping during the maneuver provided better results than gliding when executing smooth descending maneuvers (i.e., *Indoors1* and *Indoors2* experiments) as the ornithopter is more responsive at controlling the altitude. Although, the horizontal flights executed in *Indoors3* experiments were the most challenging, the reported average goal position error was 0.40 m, which is consistent with other state-of-the-art methods based on external perception systems such as [6].

VI. Conclusions and Future Work

This paper presented the first control method for bird-scale flapping-wing robots that closes the loop using a fully onboard perception system. The proposed approach exploits the advantages of event-based vision to guide an ornithopter robot towards a goal. Our scheme includes three main modules. First, an event-based line tracker provides fast-response and robust line estimations during the maneuver by fusing both *event images* and *event-by-event* processing. Second, an efficient visual servoing approach guides an ornithopter robot to match the current and goal features in the event camera plane. Third, a control system actuates over the horizontal and vertical tail deflections during the flight maneuver. The scheme has been validated online and executed on board a real ornithopter robot that equips low-cost processing hardware. It has been experimentally validated with different maneuvers in a number of indoor and outdoor scenarios.

Future work includes developing novel methods to endow ornithopter robots with the necessary capabilities to perform other challenging maneuvers such as obstacle avoidance, landing, and perching.

References

[1] G. de Croon, "Flapping wing drones show off their skills," *Sci. Robot.*, vol. 5, 2020.

[2] N. Haider, A. Shahzad, M. N. Mumtaz Qadri, and S. I. Ali Shah, "Recent progress in flapping wings for micro aerial vehicle applications," *Proc. of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 2020.

[3] E. Farrell Helbling and R. J. Wood, "A review of propulsion, power, and control architectures for insect-scale flapping-wing vehicles," *Applied Mechanics Reviews*, vol. 70, no. 1, 2018.

[4] S. Tijmons, G. C. de Croon, B. D. Remes, C. De Wagter, and M. Mulder, "Obstacle avoidance strategy using onboard stereo vision on a flapping wing mav," *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 858–874, 2017.

[5] A. Gómez Eguíluz, J. P. Rodríguez-Gómez, J. Paneque, P. Grau, J. R. Martinez De-Dios, and A. Ollero, "Towards flapping wing robot visual perception: Opportunities and challenges," in *IEEE RED-UAS*, 2019.

[6] F. Maldonado, J. Á. Acosta, J. Tormo-Barbero, P. Grau, M. Guzmán, and A. Ollero, "Adaptive nonlinear control for perching of a bioinspired ornithopter," in *IEEE/RSJ IROS*, 2020.

[7] R. Zufferey, J. Tormo-Barbero, M. Guzman, F. Maldonado, E. Sanchez-Laulhe, P. Grau, M. Perez, J. Á. Acosta, and A. Ollero, "Design of the high-payload flapping wing robot e-flap," *IEEE Robotics and Automation Letters*, 2021.

[8] F.-Y. Hsiao, L.-J. Yang, S.-H. Lin, C.-L. Chen, and J.-F. Shen, "Autopilots for ultra lightweight robotic birds: Automatic altitude control and system integration of a sub-10 g weight flapping-wing micro air vehicle," *IEEE Control Systems Magazine*, vol. 32, no. 5, pp. 35–48, 2012.

[9] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-based vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[10] A. Gómez Eguíluz, J. P. Rodríguez-Gómez, J. R. Martínez-de Dios, and A. Ollero, "Asynchronous event-based line tracking for time-to-contact manuevers in UAS," in *IEEE/RSJ IROS*, 2020.

[11] J. P. Rodríguez-Gómez, A. Gómez Eguíluz, J. R. Martinez De-Dios, and A. Ollero, "ROSS-LAN: RObotic Sensing Simulation scheme for bioinspired robotic bird LANding," in *ROBOT*. Springer, 2019.

[12] E. Pan, X. Liang, and W. Xu, "Development of vision stabilizing system for a large-scale flapping-wing robotic bird," *IEEE Sensors Journal*, 2020.

[13] A. Mitrokhin, C. Fermüller, C. Parameshwara, and Y. Aloimonos, "Event-based moving object detection and tracking," in *IEEE/RSJ IROS*, 2018, pp. 1–9.

[14] B. J. Pijnacker Hordijk, K. Y. Scheper, and G. C. De Croon, "Vertical landing for micro air vehicles using event-based optical flow," *Journal of Field Robotics*, vol. 35, no. 1, pp. 69–90, 2018.

[15] N. J. Sanket, C. M. Parameshwara, C. D. Singh, A. V. Kuruttukulam, C. Fermuller, D. Scaramuzza, and Y. Aloimonos, "Evdodgenet: Deep dynamic obstacle dodging with event cameras," in *IEEE ICRA*, 2020, pp. 10 651–10 657.

[16] D. Falanga, K. Kleber, and D. Scaramuzza, "Dynamic obstacle avoidance for quadrotors with event cameras," *Science Robotics*, vol. 5, no. 40, 2020.

[17] S. Sun, G. Cioffi, C. de Visser, and D. Scaramuzza, "Autonomous quadrotor flight despite rotor failure with onboard vision sensors: Frames vs. events," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 580–587, 2021.

[18] R. Li, D. Shi, Y. Zhang, K. Li, and R. Li, "Fa-harris: A fast and asynchronous corner detector for event cameras," in *IEEE/RSJ IROS*, 2019.

[19] I. Alzugaray and M. Chli, "Asynchronous corner detection and tracking for event cameras in real time," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3177–3184, 2018.

[20] J. P. Rodríguez-Gómez, A. Gómez Eguíluz, J. R. Martínez-de Dios, and A. Ollero, "Asynchronous event-based clustering and tracking for intrusion monitoring in UAS," in *IEEE ICRA*, 2020.

[21] E. Mueggler, B. Huber, and D. Scaramuzza, "Event-based, 6-DOF pose tracking for high-speed maneuvers," in *IEEE/RSJ IROS*, 2014, pp. 2761–2768.

[22] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-based visual inertial odometry," in *2017 IEEE CVPR*, 2017, pp. 5816–5824.

[23] R. S. Dimitrova, M. Gehrig, D. Brescianini, and D. Scaramuzza, "Towards low-latency high-bandwidth control of quadrotors using event cameras," in *IEEE ICRA*, 2020, pp. 4294–4300.

[24] J. Rodríguez Gómez, R. Tapia, J. Paneque, A. Gómez Eguíluz, P. Grau, J. Martínez de Dios, and A. Ollero, "The GRIFFIN perception dataset: Bridging the gap between flapping-wing flight and robotic perception," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1066–1073, April 2021.

[25] R. Tapia, A. Gómez Eguíluz, J. Martınez-de Dios, and A. Ollero, "ASAP: Adaptive scheme for asynchronous processing of event-based vision algorithms," in *IEEE ICRA Workshop on Unconventional Sensors in Robotics*, 2020.

[26] H. K. Khalil, *Nonlinear control*. Pearson, 2014, vol. 1.

[27] B. Siciliano and O. Khatib, *Springer handbook of robotics*, 2016.

[28] H. Jabbari, G. Oriolo, and H. Bolandi, "An adaptive scheme for image-based visual servoing of an underactuated uav," *Int. Journal of Robotics and Automation*, vol. 29, no. 1, pp. 92–104, 2014.

[29] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. USA: Cambridge University Press, 2003.

[30] N. K. Sinha and N. Ananthkrishnan, *Advanced flight dynamics with elements of flight control*. CRC Press, 2017.