2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Asynchronous Event-based Line Tracking for Time-to-Contact Maneuvers in UAS

A. Gómez Eguíluz, J.P. Rodríguez-Gómez, J.R. Martínez-de Dios and A. Ollero

***Abstract*— This paper presents an bio-inspired event-based perception scheme for agile aerial robot maneuvering. It tries to mimic birds, which perform purposeful maneuvers by closing the separation in the retinal image (w.r.t. the goal) to follow time-to-contact trajectories. The proposed approach is based on event cameras, also called artificial retinas, which provide fast response and robustness against motion blur and lighting conditions. Our scheme guides the robot by only adjusting the position of features extracted in the event image plane to their goal positions at a predefined time using smooth time-to-contact trajectories. The proposed scheme is robust, efficient and can be added on top of commonly-used aerial robot velocity controllers. It has been validated on-board a UAV with real-time computation in low-cost hardware during sets of experiments with different descent maneuvers and lighting conditions.**

***Index Terms*— event camera, aerial robots, perception systems, visual servoing, IBVS, tau theory, feature tracking.**

I. Introduction

The new advances in aerial robot technology have motivated intense research effort in performing faster and more agile maneuvers. Agile aerial robot maneuvers face relevant perception problems. First, the perception systems should be efficient enough to provide the required high control rates, often constraining the applicability of most planning-based robot navigation approaches. Image-based visual servoing methods directly use measurements obtained in the image plane to efficiently guide the robot. They provide remarkable results [1] [2] but are limited by the intrinsic nature of visual cameras, which are sensitive to lighting conditions and are severely affected by motion blur originated by the robot motion or vibrations. In this paper we use event cameras, which provide high dynamic range and temporal resolution, being insensitive to motion blur. A good number of successful techniques have been proposed in the last years evidencing their suitability, see e.g. [3]. The advantages of event cameras in many applications have attracted the manufacturers' interest causing a decrease in the camera weight and cost. We are interested in perception-based methods designed to be executed with low-cost hardware on top of standard velocity aerial robot controllers without significant modifications. Although many successful control methods have obtained outstanding results for aggressive maneuvers, see e.g. [1] [4], they are out of the scope of this paper.

This paper presents an event-based perception scheme for agile aerial robot maneuvers. It has a strong biological inspiration and tries to mimic birds, which perform perching and landing by matching the retinal separation of the images of their feet and the goal following time-to-contact trajectories [5], [6]. Event cameras mimic biological retinas in performing fast-response asynchronous data-driven light acquisition with high insensitivity to motion blur and lighting conditions. Besides, our scheme guides the robot using Tau theory by adjusting the position of features in the event image plane to their goal positions at a predefined time using smooth time-to-contact trajectories. This work has been developed in the context of the ERC Advanced Grant GRIFFIN project, which aims to develop aerial robots capable of navigating, perching and manipulating objects.

The proposed scheme guides the robot to perform Tau theory trajectories using event-based line features, which are caused by objects frequently found in many scenarios such as cables, pipes, windows, among many others, and can be more robustly detected than event-based punctual features such as corners. The scheme includes two main modules. First, straight lines are efficiently tracked combining fast-response event-by-event line tracking with robust event-image line extraction. The second module uses the tracked lines in the event camera plane to directly compute the velocity commands to the robot low-level controller such that the tracked lines describe the desired time-to-contact trajectories in the image plane. The scheme is robust, efficient and can be added on top of commonly-used aerial robot velocity controllers. It has been validated with on-board real-time computation in low-cost hardware in sets of experiments with different types of descent maneuvers and lighting conditions.

The rest of the paper is organized as follows. Section II briefly summarizes the main works in the topics addressed in the paper. The event-based line tracker and Tau guiding methods are presented in Sections III and IV, respectively. Section V shows the experimental validation. Section VI concludes the paper and highlights future research.

II. State of the art

The biological Tau-theory [7] states that humans and animals, specially birds, rely on the time-to-contact to guide most of their purposeful movements. There is evidence that pigeons adjust the breaking during landing and perching keeping constant the rate of change of τ [5], i.e. an approximation of the time-to-contact. It has been postulated that

The authors gratefully acknowledge the support at code debugging of Julio L. Paneque (GRVC Robotics laboratory). This work was supported by the European Research Council as part of GRIFFIN ERC Advanced Grant 2017 (Action 788247), the European Commission as part of AERIAL-CORE project (H2020-2019-87147), and ARM-EXTEND (DPI2017-8979-R) project funded by the Spanish National R&D Plan.
The authors are with the GRVC Robotics laboratory, University of Seville, Seville 41092, Spain email: {ageguiluz, jprodriguez, jdedios, aollero}@us.es

animals do not require cognitive processing for Tau-theory guidance, it is available at neural circuit level [7].

Recently, a number of works have explored Tau-theory for performing motions that close an action-gap (i.e the separation between the current state and a goal state), by controlling the time-to-closure τ at the current closure rate. Tau-theory has been used to endow robots with guidance capabilities [8] and to perform short-distance maneuvers such as breaking [9], landing [10] and perching [11]. Works [8], [9] employed Tau-theory to guide the movement of multi-rotor Unmanned Aerial Vehicle (UAV) providing remarkable results for autonomous navigation, docking and landing.

Two different approaches can be distinguished in Tau-theory based guiding, Tau-dot methods and intrinsic Tau guidance methods. Tau-dot methods are used for smoothly decelerate to zero contact velocity. A number of works have approached Tau-dot strategy for unmanned aerial [9] [11] and ground vehicles [12]. Tau-dot strategy has also been used for simulating bird-like landing and perching maneuvers in order to provide sensor measurements [13]. The work in [11] extended Tau-dot guidance strategy to close a gap with non-zero final velocity with a rotary-wing MAV. The gap closure was approached in three stages, acceleration, deceleration and constant velocity tracking the reference τ while keeping the perch within the field of view.

Our work falls within the so-called intrinsic-Tau guidance, which computes the robot trajectories starting from an initial state with zero velocity and acceleration, and closing the gap with zero final velocity and acceleration in a predefined time interval, see e.g. [6], [14]. Additionally, different works improved standard intrinsic-Tau guidance systems by including non-zero initial velocity such as [10], [15]. Other works have explored intrinsic-Tau guidance during the planning stage. The work [14] proposed an intrinsic-Tau guidance method for multi-UAV collision avoidance detecting collisions between trajectories obtained using intrinsic-Tau guidance, which are then re-planned using particle swarm optimization.

Most of the existing intrinsic-Tau methods guide the robot closing the gap defined in space, which requires transforming from image to space, which could affect guiding command rate in fast maneuvers. The work in [11] use a vision system build on top of an attitude controller to control the motion of a MAV during the landing/perching maneuvers. However, the method does not rely only on visual input and suffers from the limitations of traditional cameras such as motion blur or sensitivity to lighting conditions. Although not relying on Tau theory strategies, a solution for the vertical landing of UAVs was proposed in [16] using the optical flow obtained from event-based vision, which overcomes the problems of using traditional cameras.

The advantages of event cameras have motivated increasing research interest of the robotics and computer vision communities [3] [17]. A wide variety of methods have been used for feature extraction [18], clustering [19], tracking [20], optical flow computation [21], detect objects in motion [22], and SLAM [23], among many others. A full review of event processing methods can be found in [3]. Most existing event processing methods group the temporally-close received events in frames called *event images*. Event cameras provide high noise level and processing event images allows reducing noise impact. However, event-image processing does not fully exploit the sequential and asynchronous nature of event cameras and some methods (e.g. [24]) require including mechanisms for cancelling motion blur.

Line detection using event cameras is a suitable alternative to extract scene structure from the event stream. Lines from a square are tracked in [25] as reference to estimate the pose of a flying drone using the Hough transform [26]. Inspired on the line segment detector LSD [27] an event-based approach detects and tracks lines by clustering events [28]. Line detection is performed in [29] using event-based visual flow and least squares. The method also provides segment detection by estimating the endpoint of the detected lines. Events are clustered in [30] to generate planes on the spatio-temporal space that define possible lines, but as reported, the method cannot track spinning lines. Although these methods provide highly accurate line detection in their experiments, they do not focus on line tracking nor on the robustness in complex and unstructured scenarios, and few of them have been validated onboard robots.

This paper presents a bio-inspired perception-based method for aerial robot maneuvers. The contribution of this method is two-fold. First, we present a robust and efficient event-based line tracking method that fuses event-by-event processing together with event-image processing, combining the fast-responsivity of the former and the robustness of the later. Second, an efficient intrinsic-Tau guidance method that commands the robot using only features in the event camera plane by adjusting them to match a set of reference features. The scheme is designed to be executed on low-cost hardware on top of standard aerial robot controllers.

III. Asynchronous Event-based Line Tracking

We are interested in perception techniques for guiding aerial robots in agile maneuvers consisting in moving the robot from its initial pose to a goal pose. The robot goal pose is defined w.r.t. a reference pattern that can be observed with no occlusions during the full maneuver. The reference pattern can be off-line predefined, e.g. the robot landing pad in landing experiment, or can be online selected during the robot navigation, e.g. to facilitate moving to a waypoint, the navigation system selects as reference a pattern near the waypoint. We assume that the reference pattern is defined by a set of straight line segments. Lines are general features that have been largely exploited in robot perception, see e.g. [31]. In our problem, a great variety of objects and scenarios can originate lines in the event camera and, many of these objects are actually used for a number of aerial robot maneuvers such as landing or perching. Line features contain richer structure and can be more robustly extracted than punctual features such as corners.

This module tracks a set of straight lines in the image plane using event processing. The events from the Dynamic Vision Sensor (DVS) of an event camera onboard the

robot are received asynchronously with μs resolution. We adopted an approach consisting in combining two types of event-based processing with different temporal resolutions. First, an event-by-event processing module analyzes every new event received and provides fast-response, but noisy, line tracking estimations. Second, an event-frame processing module accumulates the N last events received and provides robust line tracking estimations. The final estimation combines both modules using an efficient Extended Kalman Filter (EKF) for each line tracked, providing fast-response and robust line tracking. The line estimations from the event-by-event processing are used in the EKF Prediction stage and, those from the event-frame processing are used in the Update stage. The EKF covariance matrices were set to assign to the observations double reliability than the predictions. To ensure the statistical consistency of the EKF, event-by-event processing and event-image processing use different events. The events received from the camera are randomly sampled as in [22]. γ_e and γ_i are the percentages of the input events that are sent to the event-by-event and the image-event processing modules, respectively. Setting γ_e and γ_i can be used to adjust the computational cost for running on real time adapting to the scenario particularities and the onboard hardware.

Each event is defined by $\mathbf{e} = (t, u, v, p)$, where t is the time in which the event was triggered, (u, v) are the pixel coordinates and p is the polarity of the brightness change, i.e. either 1 or 0. Both event processing methods are based on the Hough space representation [26]. In polar coordinates, lines are expressed as $\rho = u\cos\theta + v\sin\theta$. Similarly, lines are defined by $l = (\theta, \rho)$ in the Hough space. To reduce the computational effort, the (θ, ρ) Hough space is discretized.

A. *Event-by-event line tracker*

The proposed event-by-event method is partially inspired by the method described in [25] and includes mechanisms to increase robustness without significantly increasing the computational burden. The method is initialized with the set L of lines that represent the reference pattern to track. Each line $l \in L$ is defined by a tuple (θ_l, ρ_l) in the Hough space and by a set P_l of N_P prototype events well distributed over l in the image plane. For each line l, buffer B_l is used to keep the actual and last prototype events representing l. To prevent over-representation, the number of elements in B_l corresponding to the same prototype event is bounded.

P_l is initialized as follows. First, the consistency of each received event $e = (u, v)$ with line l is evaluated in the Hough space. The representation of e in the Hough space is (θ_l, ρ_e), where $\rho_e = u\cos\theta_l + v\sin\theta_l$. If the distance between (θ_l, ρ_l) and (θ_l, ρ_e) is low, e is assigned as belonging to line l. Next, to ensure a good distribution of prototype events, e is added to P_l if the distance to its nearest element in P_l is greater than a threshold. This procedure is iterated until P_l contains N_P prototype events. The events in P_l are buffered in B_l and projected into a Hough space of events. The parameters of the line (θ_l, ρ_l) are computed as the centroid of the event Hough space originated by B_l. Hence, each line l is characterized by (θ_l, ρ_l), P_l, B_l and, for computational reasons, the event Hough space of B_l.

After initialization, the event-by-event line tracker processes every new event e as follows. First, the consistency of e with every line l is evaluated as described above. If consistent with only one line l, e is assigned to l. Events consistent with several lines are discarded to avoid contributions from line intersections. Events not assigned to any line are discarded. Next, if the distance between e and the nearest prototype event in P_l is higher than a distance, e is taken as spurious and is discarded. In order to enable line updates with new events, if the distance between e and the nearest prototype event in P_l is below a distance, P_l and B_l are updated: the prototype event in P_l is substituted by e; and e is added to B_l eventually deleting the oldest event in B_l in case the number events corresponding to that prototype exceeds the bound. The operation in one example is shown in Figure 1.

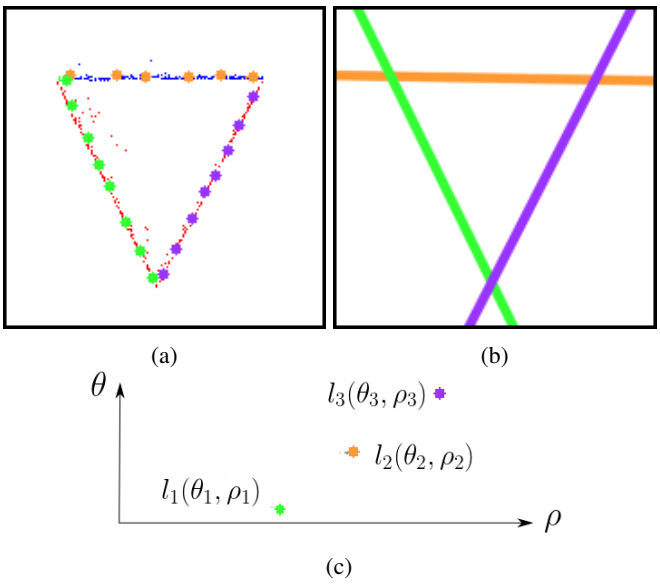


Fig. 1: Operation of event-by-event line tracking in one example: a) input events received asynchronously and processed –small red and blue dots– and events in P_l –big dots– for the three tracked lines; b) tracked lines in the image plane; c) zoomed event Hough space showing the tracked lines.

The updated line parameters (θ_l, ρ_l) are computed as the coordinates of the centroid of the event Hough space originated by the elements in B_l, see Figure 1c. This computation is efficient: the contribution of the new element in B_l is added to the event Hough space and, the contribution of the deleted element, removed. Finally, old elements of P_l and B_l are deleted by using their timestamp. Notice that old element check and deletion are also efficient due to the asynchronous nature of events.

The operation of the method is summarized in Alg. 1. For efficiency, it uses the Manhattan distance. Although performs event-by-event processing, it has low computational cost suitable for onboard execution in low-cost hardware.

Algorithm 1: Asynchronous event-by-event line tracking

Input: e, L
Output: L
$candidates \leftarrow 0$
for $k \leftarrow 1$ *to* $length(L)$ **do**
  if *belongToLine(e,k)* **then**
    $candidates \leftarrow candidates + 1$
    $l \leftarrow k$
  end
end
$association \leftarrow False$
if $candidates = 1$ **then**
  $P_l = getPrototypeList(l)$
  if *nearToAnyPrototype(e, P_l)* **then**
    $replacePrototype(e, P_l)$
    $association \leftarrow True$
  else if $\neg\ isFull(P_l)$ **then**
    if *fitsIn(e,P_l)* **then**
      $addPrototype(e, P_l)$
      $association \leftarrow True$
  if $association$ **then**
    $updateHoughSpace(e)$
    $ekfPrediction(l)$
  end
end

B. Event-frame line tracker

This module extracts lines in event frames generated by accumulating the last received events. This approach was preferred over using frames with events accumulated in a fixed time window since it naturally adapts the event frame generation to the relative camera-scene motion: highly dynamic scenes, which require faster response, trigger more event-frames than static scenes. The method is efficient and can be executed online and onboard with low-cost hardware.

This method extracts lines from the event-image and associates them to the tracked lines. First, the Hough transform is computed over the event image. Due to the Hough space discretization, each line in the image plane provides contributions to nearby locations. The centroids in the Hough space are used to represent lines. Finally, these candidate lines are associated to the tracked lines using minimum Hough-space distance. The EKF is updated using the new extracted lines as observation.

IV. Time-to-Contact Velocity Guidance

This section presents a Tau-theory based method that uses the lines tracked in the image plane to directly compute the velocity commands to the robot such that the tracked lines describe the desired time-to-contact (TTC) trajectories in the image plane. We adopt an intrinsic-Tau guidance approach that closes the gap to the goal in a predefined time and with zero final velocity and acceleration. Unlike most existing intrinsic-Tau methods used in robotics, the gap is defined directly in the image plane, which enables fast response. In our problem each gap is defined by the difference between the actual features (lines extracted from the event camera) and the goal features in image plane. Assuming calibrated cameras and the target geometry being known, the position of the goal features in the image plane can be easily computed by projecting the reference pattern as if the robot were at the goal pose. The computation of goal features is performed only once before the maneuver starts. Closing the gaps will drive the image features towards their goal positions, which will result in the robot reaching the goal pose.

For clarity, we first describe the adopted intrinsic-Tau guidance strategy for one only gap. For a given gap $\chi(t)$, $\tau(t)$ provides a first order approximation of the time-to-contact computed as $\tau(t) = \chi(t)/\dot{\chi}(t)$, where $\dot{\chi}(t)$ is the gap closing rate at time t. By convention, $\chi(t)$ is always negative and the gap is closing when $\dot{\chi}(t)$ is positive.

For each feature at time t we can define the gap distance $\chi(t)$, its closure velocity rate $\dot{\chi}(t)$, and its closure acceleration $\ddot{\chi}(t)$. We focus in maneuvers that start at rest (i.e. $\chi(0) \neq 0, \dot{\chi}(0) = \ddot{\chi}(0) = 0$), accelerate to a peak velocity and then, decelerate to end the movement at rest at time T_g ($\chi(T_g) = \dot{\chi}(T_g) = \ddot{\chi}(T_g) = 0$). These type of motions are of interest for many time-constrained maneuvers such as landing and perching. As reported in [9], the above intrinsic Tau-guidance trajectory can be computed as follows:

$$\begin{aligned}
\chi(t) &= \frac{\chi}{T_g^{2/k}}\left(T_g^2 - t^2\right)^{\frac{1}{k}},\\
\dot{\chi}(t) &= -2t\frac{\chi_{(0)}}{kT_g^{2/k}}\left(T_g^2 - t^2\right)^{\frac{1}{k}},\\
\ddot{\chi}(t) &= -2t\frac{\chi_{(0)}}{kT_g^{2/k}}\left[\frac{2-k}{k}t^2 - T_g^2\right]\left(T_g^2 - t^2\right)^{\frac{1-2k}{k}},
\end{aligned} \tag{1}$$

where k determines the trajectory kinematics. In particular, selecting $k \in (0, 0.5]$ ensures $\chi(T_g) = \dot{\chi}(T_g) = \ddot{\chi}(T_g) = 0$.

We aim at closing the gaps for all features altogether by using the concept of Tau-coupling [7]. One gap is declared as main, typically that with the greatest gap distance at $t = 0$, and the rest are declared as coupled. The evolution of the main gap is computed with Eq. (1). Gap coupling is characterized by κ, a constant that expresses the relation between both gap closing rates such as $\tau_\chi = \kappa\tau_\zeta$. Therefore, the trajectory of every coupled gap is obtained as follows:

$$\begin{aligned}
\zeta(t) &= c\chi(t)^{\frac{1}{\kappa}-1},\\
\dot{\zeta}(t) &= c\frac{1}{\kappa}\dot{\chi}(t)\chi(t)^{\frac{1}{\kappa}-1},\\
\ddot{\zeta}(t) &= c\frac{1}{\kappa}\left(\left(\frac{1}{k}-1\right)\dot{\chi}(t)^2 + \chi(t) + \ddot{\chi}(t)\right)\chi(t)^{\frac{1}{\kappa}-2},
\end{aligned} \tag{2}$$

where $c = \zeta(0)/\chi(0)^{(1/\kappa)}$ establishes the relation between the main and the coupled gap.

Next, the robot velocity commands are computed using Image-Based Visual Servoing (IBVS). As described above, $\chi(t)$ and $\zeta(t)$ describe the desired gap distances to accomplish the intrinsic-Tau guidance maneuver: they are the reference for our controller. Also, we can obtain the actual

gap distances at a given time t by using the lines tracked as in Section III and the goal lines. Let $e(t)$ be a vector with the difference between the actual gap distances and the reference gap distances at time t. Each feature defines a number of gaps, one for each feature parameter. In particular, each tracked line defines two gaps, one in θ and one in ρ. Hence, each tracked line creates two entries in $e(t)$: the error in ρ using the Euclidean distance, and the error in θ using the circular distance. We adopted a simple control law (see [32]) to compute the camera velocities $\nu(t)$ at time t:

$$\nu(t) = -KJ^*e(t) + J^*\dot{e}(t), \tag{3}$$

where K is a positive definite diagonal weighing matrix, $\dot{e}(t)$ is the error between the current gap closure velocities and their reference values obtained using Eqs. (1) and (2), and J^* is the pseudoinverse of J, the Jacobian that describes the variation of the gap distance as a function of camera velocity, computed as shown below. We preferred to use a control law with a derivative term in order to improve the transient response and anticipate error cancellation. Notice that $\nu(t)$ is expressed in the camera frame and has to be transformed to the robot frame.

J is obtained as follows. Assuming features obtained with normalized focal length, i.e. $f = 1$, the kinematics of line l in the image plane can be expressed as $\dot{l} = J_l\nu$, where ν is the camera (linear and angular) velocity vector and J_l is the Jacobian that describes the variation of the line parameters as a function of camera velocity. J_l can be computed as:

$$J_l = \begin{bmatrix} \lambda_\theta s(\theta) & \lambda_\theta c(\theta) & -\rho\lambda_\theta & -\rho s(\theta) & -\rho c(\theta) & -1 \\ \lambda_\rho s(\theta) & \lambda_\rho c(\theta) & -\rho\lambda_\rho & -c(\theta)(1+\rho^2) & s(\theta)(1+\rho^2) & 0 \end{bmatrix} \tag{4}$$

where $s(\cdot)$ and $c(\cdot)$ stand for sine and cosine and

$$\begin{aligned} \lambda_\theta &= (\arccos\theta - bs(\theta))/d, \\ \lambda_\rho &= -(a\rho s(\theta) + b\rho c(\theta) + c)/d, \end{aligned} \tag{5}$$

where $aX + bY + cZ + d = 0$ defines the plane in space that contains the line. Although quite tolerant to a certain degree of error in the plane parameters, a broad estimate of the line depth is required. We compute the depth by derotating the optical flow at the intersections between the tracked lines as described in [33].

The gaps are defined by the lines extracted in the image plane and the goal lines. Only the current line features are affected by the camera velocities while the goal lines are constant. Hence, J_l, the Jacobian of a line, represents the variation of the gap distance with the camera velocity. J can be computed by concatenating row-wise the Jacobians J_l for all tracked lines. Each line contributes with two rows in J, one for θ and one for ρ. A minimum of three non-colinear lines is required to have a full rank J. However, higher number of lines enhances accuracy and prevents potential temporary co-linearity between lines along the maneuver.

Standard multirotors are underactuated and require tilting in the direction they desire to translate. To compensate for these rotations the image coordinates are re-projected into a virtual image plane as in [2]. The robot roll and pitch angles (ϕ and φ) are estimated using an Inertial Measurement Unit (IMU). Assuming null translation in the x and y axes between the camera and robot coordinate frames, the re-projection of image coordinates (u, v) into the virtual image plane ϑ can be computed as:

$$\begin{bmatrix} u_\vartheta \\ v_\vartheta \\ f \end{bmatrix} = \beta R_{\phi\varphi} \begin{bmatrix} u \\ v \\ f \end{bmatrix}, \tag{6}$$

where $R_{\phi\varphi}$ is the rotation matrix associated to the robot roll and pitch angles, and β ensures the re-projection in an image plane with focal length f using:

$$\beta = f / \left([0\ 0\ 1] R_{\phi\varphi} \begin{bmatrix} u \\ v \\ f \end{bmatrix} \right) \tag{7}$$

Re-projection could not be necessary when the robot moves slowly or is endowed with a non-underactuated controller such as [34]. However, our scheme is designed to be valid on most multicopters, which are typically underactuated.

V. Experimental Results

The proposed scheme has been validated in sets of experiments performing different descent maneuvers and with different lighting conditions. The aerial platform includes a *DJI Flamewheel F450* frame with a *PixRacer* autopilot. A DAVIS 346 event camera was mounted pointing downwards (-90^o pitch rotation) and a low-cost *Khadas VIM3* board is used for logging and real-time computation, see Figure 2. The method was implemented in C++ on top of the UAL abstraction layer [35] using *ROS Kinetic* and the ASAP event processing scheme [36]. The output of the proposed method was used as input of the default *PX4* velocity-based low-level controller. The experiments were performed in a testbed equipped with 24 *OptiTrack Prime^{x}13* cameras that provided millimeter accuracy robot pose estimations, used only as ground truth for evaluation.

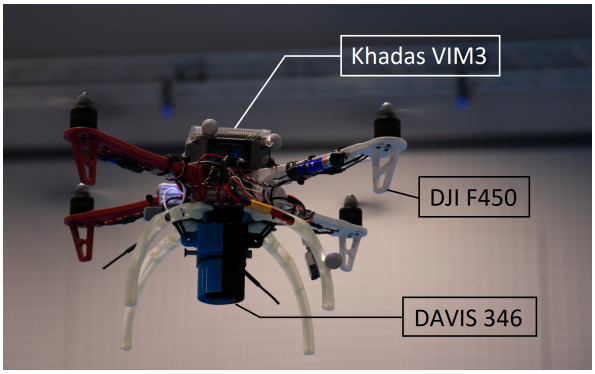


Fig. 2: Aerial robot used in the experiments.

A. Preliminary Experiments

First, we evaluated the accuracy and robustness of the line tracking method and confirmed the suitability of its execution rates for robot guiding. The performance of the line tracker was robust to reasonable values of its parameters. The size

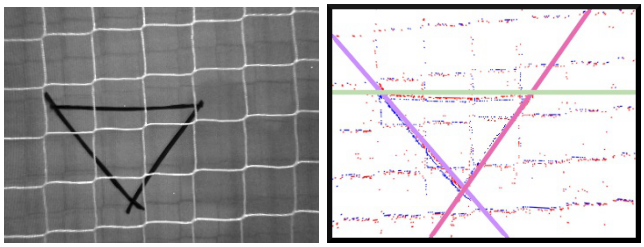


Fig. 3: Visual image and event image with the tracked lines taken in the safety net scenario.

of P_l was chosen as 6 and, the size of B_l, as 20. Further, γ_e and γ_i were set to 0.6 and 0.4 respectively to enhance computation without losing robustness. For these experiments, event images were obtained by accumulating 2,000 events as a trade-off between robustness and computational effort.

The accuracy of the line tracking method was evaluated by measuring the error between the resulting tracked lines and the ground truth (GT) lines extracted applying the well-known Canny edge detector on the visual images provided by the *Davis APS* sensor. The tracking error was measured as the mean distance between the pixels detected by Canny as belonging to the GT lines and the lines tracked by our method. We want to asses the accuracy at line tracking, not the line detection capabilities. Our line tracking method was initialized with the GT lines and during the rest of the experiment, it was fed only with events from the *Davis DVS* sensor as described in Section III. The mean error, estimated averaging more than 20,000 different line tracking, was lower than 2.23 pixels, which was sufficient for our scheme.

Next, we evaluate the robustness of our method in highly cluttered environments, when different objects partially occlude the tracked lines: *flying objects*, *office background* and the *testbed safety net*. Figure 3 shows a visual image and one event image with the tracked lines taken from the experiment in the safety net scenario, which is particularly hard since the net originates many line-distributed spurious events. Table I shows the duration of the cluttered scenes and the lifetime of the lines tracked. In all cases the lines were robustly tracked.

TABLE I: Robustness of line tracking lifetimes with different cluttered environments.

	Duration (s)	**LifeTime (s)**	%
Flying Object	15.5	14.82	96
Office	35.06	32.02	91
Protection Net	18.48	14.58	78

Additionally, we evaluate the execution rates performed by our proposed line tracking method running onboard the *Khadas VIM3* and, compared it versus two other line extractors. *Method1* applies the Hough transform on visual images from the DAVIS APS sensor. *Method2* extracts lines from event images. The three methods were executed for the same experiment simultaneously tracking 4 lines. Table II shows the execution rates of the line tracker and the resulting command rates of the full guiding system. Running in the low-cost *Khadas VIM3* board, the proposed line tracking method provided average execution rates of $348Hz$ enabling control command rates of $339Hz$, both significantly higher than those of *Method1* and *Method2*. It is worth noting that the execution times of *Method2* and the proposed approach are not constant (i.e. they are asynchronous) and these values are average values from our experiments. Therefore, their performance depends on the scene dynamics and can reach frequencies significantly higher at times while providing lower rates when the scene is static. However, these rates were valid to perform all the robot guiding experiments. Additionally, we empirically evaluate the scalability of the system and conclude that the computational cost of both algorithms (i.e. line extraction and TTC guidance) grows linearly with the numbers of lines. Although using a very high number of lines reduces the performance of the proposed method, one can choose the number of lines to track beforehand in order to provide the sufficient command rate required for the desired maneuver.

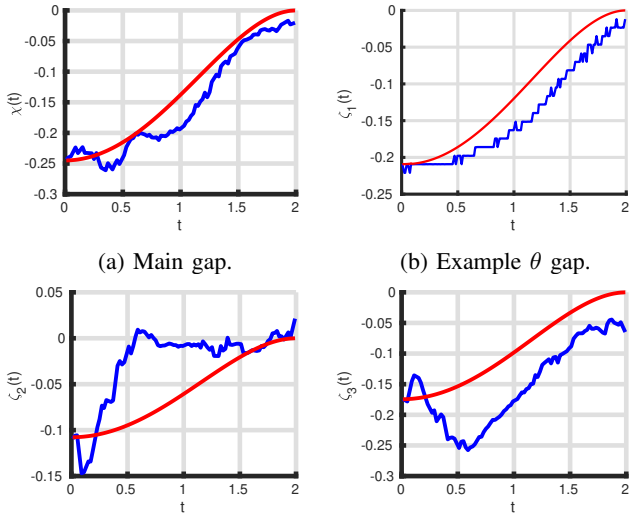
TABLE II: Execution times and control command rates of *Method1*, *Method2* and the adopted line tracker.

	Method1	***Method2***	**Adopted method**
Line extraction	38 Hz.	68 Hz.	348 Hz.
Line extraction + TTC guidance	37 Hz.	67 Hz.	339 Hz.

B. Event-based Tau Guidance Experiments

A total of 180 descent maneuvers were performed covering different initial and goal poses, maneuver's duration T_g, and lighting conditions. The selected values of T_g ensured that the robot spatial velocities were lower than the low-level controller safety bounds, $2m/s$ in our case. The well-known camera retreat issues [33] of the IBVS method were prevented by selecting a rotation between the initial and goal robot poses lower than $\pi/4$. Although multiple initial and goal configurations were evaluated, all experiments were performed with the same parameters. The gain of the image-based control law is set to 0.2 in all the diagonal entries, i.e. $K = 0.2I_{6x6}$. We selected $\kappa = 1$ to ensure that coupled gaps close at the same time that the main gap. Also, $k = 0.5$, which determines the gap trajectory kinematics was set to ensure that gaps are closed with final zero velocity and acceleration.

Figure 4 shows the gap evolution along a maneuver in which the robot covered a distance of $1.5m$ in $T_g = 2s$. It is worth noting that the maneuver starts and finish at zero velocity and, therefore, it requires reaching a maximum velocity close to the safety bounds, i.e. $2m/s$. The experiment was performed with 4 lines, i.e. 8 gaps. For brevity we show only four gaps: a) the main ρ gap $\chi(t)$, b) an example θ gap $\zeta_1(t)$ and, c) and d) the ρ coupled gaps that best $\zeta_2(t)$ and worst $\zeta_3(t)$ reduced the gap at T_g. The y-axis represents the gap distance assuming unitary focal length. The gap trajectory predicted using intrinsic-Tau guidance strategy is shown in red, and the evolution of the actual gap resulting in the experiment, in blue. Closing θ gaps only involves robot rotations in Yaw angle, hence TTC trajectories can be easily



(a) Main gap. (b) Example θ gap.

(c) Best coupled ρ gap. (d) Worse coupled ρ gap.

Fig. 4: Evolution of 4 gaps along a maneuver: a) main ρ gap, b) a θ gap, c) and d) the best and worst ρ coupled gaps.

followed with low error (Figure 4b). Closing ρ gaps is not that accurate since it involves robot spatial translations in X, Y and Z axes. The main gap (Figure 4a) was closed more accurately than the coupled gaps. Although not all the ρ gaps might be fully closed at T_g, they are consistently reduced in a smooth way within the time constrains. The average ρ and θ gap closing error at T_g were 0.32 pixels and 0.5^o in the event camera image plane. In the maneuver the robot follows a smooth spatial evolution. Figure 5 shows the 3D trajectory followed by the robot and the distance between the robot and the goal position along the trajectory. After T_g the robot pose had an error (Euclidean distance) over the goal robot pose of $10.4cm$, measured with the *OptiTrack*. Similar performance was also noticed in all the maneuvers performed.

The error in reaching a robot goal pose depends on the duration of the maneuver. Figure 6 shows the average robot error when performing the same trajectory using different of $T_g \in [2, 10]s$. The errors were obtained averaging the results in 20 maneuvers for each value of T_g. All the maneuvers were performed with the same method parameters, including

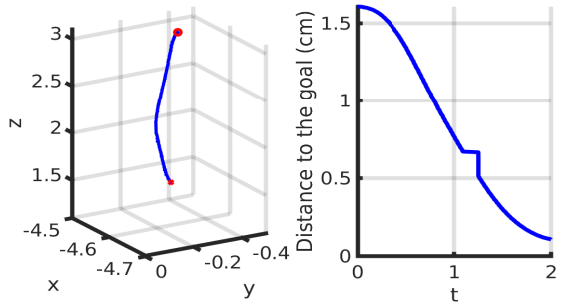


Fig. 5: Left) 3D trajectory followed by the robot. Right) Spatial robot-goal distance along the maneuver.

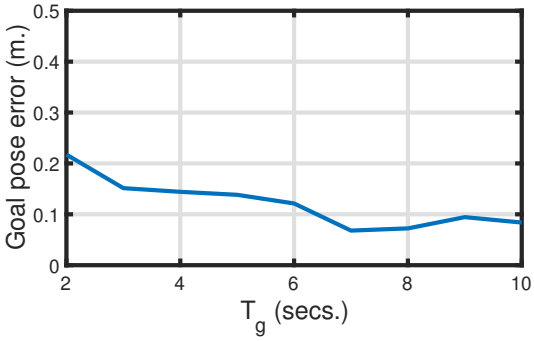
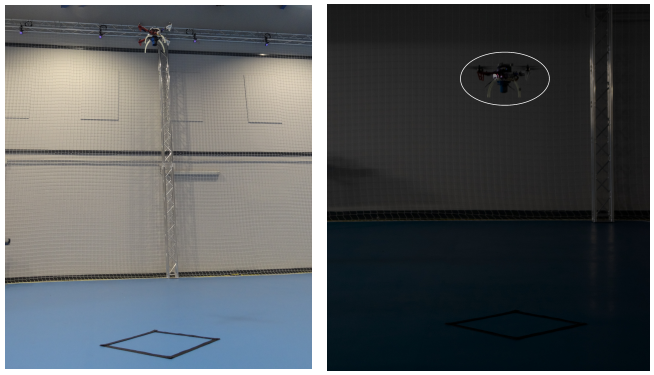


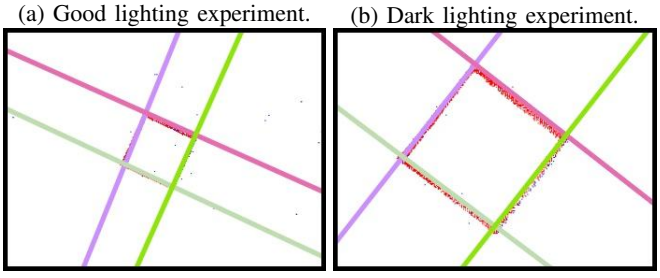
Fig. 6: Average robot spatial error in reaching the goal pose.

the control law gain K. Figure 6 shows that the error decreases as the value of T_g increases. With low values of T_g, the adopted method provides stronger velocity commands. The limitations in the responsiveness to the velocity commands of the controller (*PX4* in our case) implemented on the robot is more noticeable with low values of T_g. This error could be reduced in case of running a IBVS method after the trajectory has ended (i.e. $t > T_g$) so that the error is reduced during hovering.

Different experiments were performed for a variety of lighting conditions including, in pitch dark. Figure 7 shows two maneuvers with different lightning conditions. Figures 7a and 7b show the trajectories followed by the UAV and Figures 7c and 7d show initial and goal line features for the same trajectories of both maneuvers. Although the experiments were performed under pitch dark (see video support material), Figure 7b was edited to allow the visibility of the scene. No performance impact was detected when the



(a) Good lighting experiment. (b) Dark lighting experiment.



(c) Initial line features. (d) Goal lines features

Fig. 7: Experiments in illuminated and dark scenarios.

UAV operated in different lightning conditions, even in pitch dark scenes, which has higher spurious event rates than well-illuminated scenes.

VI. Conclusions and Future Work

This paper presents a bio-inspired perception-based method for aerial robot maneuvers. It is based on event cameras, also called artificial retinas, which provide fast response and robustness against motion blur and lighting conditions. The scheme tries to mimic birds, which perform purposeful maneuvers, such as landing and perching, by matching the separation of the retinal image and the goal, following time-to-contact trajectories.

Our scheme includes two main modules. First, a event-based line tracking method combines event-by-event and event-image processing, providing fast-response and robust line estimations. Second, an efficient intrinsic-Tau guidance method commands the robot adjusting the features in the event camera plane to a set of reference features and, thus, describing smooth TTC trajectories. The scheme has been designed to be online executed onboard low-cost hardware using standard aerial robot controllers. It has been experimentally validated with different descent maneuvers and challenging lighting conditions. The adaptation and extension of the proposed scheme to be used onboard ornithopter robots is object of future work.

References

[1] J. Thomas, G. Loianno, K. Sreenath, and V. Kumar, "Toward image based visual servoing for aerial grasping and perching," in *2014 IEEE ICRA*. IEEE, 2014, pp. 2113–2118.

[2] 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.

[3] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, *et al.*, "Event-based vision: A survey," *arXiv preprint arXiv:1904.08405*, 2019.

[4] S. Spedicato, G. Notarstefano, H. H. Bülthoff, and A. Franchi, "Aggressive maneuver regulation of a quadrotor uav," in *Robotics Research*. Springer, 2016, pp. 95–112.

[5] D. N. Lee, M. N. Davies, P. R. Green, *et al.*, "Visual control of velocity of approach by pigeons when landing," *Journal of Experimental Biology*, vol. 180, no. 1, pp. 85–104, 1993.

[6] J. M. Dietl and E. Garcia, "Ornithopter optimal trajectory control," *Aerospace Science and Technology*, vol. 26, no. 1, pp. 192–199, 2013.

[7] D. N. Lee, "General tau theory: evolution to date." *Perception*, vol. 38, no. 6, p. 837, 2009.

[8] F. Kendoul and B. Ahmed, "Bio-inspired taupilot for automated aerial 4D docking and landing of unmanned aircraft systems," in *2012 IEEE/RSJ IROS*, 2012, pp. 480–487.

[9] F. Kendoul, "Four-dimensional guidance and control of movement using time-to-contact: Application to automated docking and landing of unmanned rotorcraft systems," *The Intl. Journal of Robotics Research*, vol. 33, no. 2, pp. 237–267, 2014.

[10] A. R. Vetrella, I. Sa, M. Popović, R. Khanna, J. Nieto, G. Fasano, D. Accardo, and R. Siegwart, "Improved tau-guidance and vision-aided navigation for robust autonomous landing of uavs," in *Field and Service Robotics*. Springer, 2018, pp. 115–128.

[11] H. Zhang, B. Cheng, and J. Zhao, "Optimal trajectory generation for time-to-contact based aerial robotic perching," *Bioinspiration & biomimetics*, vol. 14, no. 1, p. 016008, 2018.

[12] ——, "Extended tau theory for robot motion control," in *2017 IEEE ICRA*, 2017, pp. 5321–5326.

[13] J. P. Rodríguez-Gómez, A. Gómez Eguíluz, J. R. Martínez-de Dios, and A. Ollero, "ROSS-LAN: Robotic sensing simulation scheme for bioinspired robotic bird landing," in *Iberian Robotics Conference*. Springer, 2019, pp. 48–59.

[14] Y. Zuqiang, F. Zhou, and L. Ping, "A bio-inspired collision-free 4D trajectory generation method for unmanned aerial vehicles based on tau theory," in *2015 34th Chinese Control Conference (CCC)*, 2015, pp. 6961–6968.

[15] Z. Yang, Z. Fang, and P. Li, "Decentralized 4D trajectory generation for uavs based on improved intrinsic tau guidance strategy," *Intl. J. of Advanced Robotic Systems*, vol. 13, no. 3, p. 88, 2016.

[16] 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.

[17] 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.

[18] I. Alzugaray and M. Chli, "Asynchronous corner detection and tracking for event cameras in real time," *IEEE RA-L*, vol. 3, no. 4, pp. 3177–3184, 2018.

[19] F. Barranco, C. Fermuller, and E. Ros, "Real-time clustering and multi-target tracking using event-based sensors," in *2018 IEEE/RSJ IROS*, 2018, pp. 5764–5769.

[20] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza, "EKLT: Asynchronous photometric feature tracking using events and frames," *Intl. Journal of Computer Vision*, pp. 1–18, 2019.

[21] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "Unsupervised event-based learning of optical flow, depth, and egomotion," in *IEEE Conf. on Comp Vision and Pattern Recognition*, 2019, pp. 989–997.

[22] 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 *2020 IEEE ICRA*. IEEE, 2020, pp. 8518–8523.

[23] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Ultimate slam? combining events, images, and imu for robust visual slam in hdr and high-speed scenarios," *IEEE RA-L*, vol. 3, no. 2, pp. 994–1001, 2018.

[24] 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 *2020 IEEE ICRA*. IEEE, 2020, pp. 10 651–10 657.

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

[26] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[27] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: A line segment detector," *Image Proc. On Line*, vol. 2, pp. 35–55, 2012.

[28] C. Brändli, J. Strubel, S. Keller, D. Scaramuzza, and T. Delbruck, "ELiSeD — an event-based line segment detector," in *IEEE Intl. Conf. on Event-based Control, Comm., and Signal Proc.*, 2016, pp. 1–7.

[29] D. R. Valeiras, X. Clady, S.-H. Ieng, and R. Benosman, "Event-based line fitting and segment detection using a neuromorphic visual sensor," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 30, no. 4, pp. 1218–1230, 2018.

[30] L. Everding and J. Conradt, "Low-latency line tracking using event-based dynamic vision sensors," *Front. in neurorobotics*, vol. 12, 2018.

[31] R. Gomez-Ojeda, F. Moreno, D. Zuñiga-Noël, D. Scaramuzza, and J. Gonzalez-Jimenez, "PL-SLAM: A stereo SLAM system through the combination of points and line segments," *IEEE Trans. on Robotics*, vol. 35, no. 3, pp. 734–746, June 2019.

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

[33] P. Corke, *Robotics, vision and control: fundamental algorithms in MATLAB®*. Springer, 2017, vol. 118.

[34] R. Carlos, J. Á. Acosta, and A. Ollero, "Command-filtered back-stepping redesign for aerial manipulators under aerodynamic and operational disturbances," in *Iberian Robotics Conference*. Springer, 2017, pp. 817–828.

[35] F. Real, A. Torres-González, P. Ramón-Soria, J. Capitán, and A. Ollero, "UAL: An abstraction layer for unmanned aerial vehicles," in *2nd Intl. Symposium on Aerial Robotics*, 2018.

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