ROSS-LAN: RObotic Sensing Simulation scheme for bioinspired robotic bird LANding

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

GRVC-Robotics Lab, University of Seville.
Seville, 41092, Spain
{jrodriguezg, ageguiluz, jdedios, aollero}@us.es

Abstract. Aerial robotics is evolving towards the design of bioinspired platforms capable of resembling the behavior of birds and insects during flight. The development of perception algorithms for navigation of ornithopters requires sensor data information to evaluate and solve the limitations presented during the flight of these platforms. However, the payload constraints and hardware complexity of ornithopters hamper the sensor data acquisition. This paper focuses on the development of a multi-sensor simulator to retrieve the sensor information captured during the landing maneuvers of ornithopters. The landing trajectory is computed by using a bioinspired trajectory generator relying on *tau theory*. Further, a dataset of the sensor information records obtained during the simulation of several landing trajectories is publicly available online[1].

Keywords: tau theory, ornithopter, event-based cameras, LiDAR

1 Introduction

The development and implementation of bioinspired aerial platforms are mainly constrained by the lack of technology and scientific knowledge to resemble the behavior of birds during flight. Currently, most of the aerial vehicles fly by using either fixed-wing or rotary-wing configurations. Differently from the previous approaches, some bioinspired vehicles such as ornithopters employ flapping-wing mechanisms to generate the thrust and lift forces necessary to fly. Although there are some advances on the development of ornithopters [1] [2] [3] [4], the design of a robotic aerial vehicle capable of resembling the flight of birds is still under development and requires research on areas such as aerodynamics, electronics, mechanics, modeling, sensing, perception, and control.

Autonomous navigation of aerial robots requires a robust and efficient perception system to ensure precise mapping, GNSS-denied pose estimation, object detection and tracking, among others. The operation of ornithopters poses strong perception challenges. The fast movements of ornithopters during flight together with the mechanical vibrations originated by the flapping motion require efficient

[1] https://grvc.us.es/bioinspired-landing-trajectory-sensor-dataset/

and robust perception methods. In most cases only one sensor is not sufficient to capture all the relevant information of the scene. Thus, testing different sensor fusion techniques is a need that contrasts with their complex validation and the limited sensor integration onboard these aerial platforms. Further, integrating hardware in ornithopters requires significant effort due to the size constraints and the weight balance requirements of the platform. The low payload limitations and the high computational burden required for ornithopter perception recommend the implementation of perception techniques in dedicated hardware (e.g. FPGAs), which involves specific hardware development. Hence, a multi-sensor simulator is a useful tool to test and select sensors, methods, and algorithms previous to their implementation and testing in real ornithopter platforms.

This paper presents a multi-sensor simulation tool that provides simulated sensor measurements for the development and evaluation of perception and sensor fusion techniques for ornithopters platforms. It includes a bioinspired trajectory generator based on *tau theory* that simulates the trajectories performed by birds during landing and perching maneuvers [5]. These are maybe the most demanding tasks from a perception perspective. For simplicity, we define the term *tau trajectory* as the trajectory computed using *tau theory*. Existing research has proven that birds use a combination of simple strategies and the value of *tau* obtained from perceptual stimuli to guide most of their intended movements. The sensory data acquisition is necessary to develop perception algorithms to integrate *tau theory* on robotic platforms. The presented sensory simulation tool receives the input parameters that define the *tau trajectory* and generates as output the measurements from the onboard sensors during the *tau trajectory* simulation. This work has been developed in the context of the ERC-GRIFFIN and ARM-EXTEND projects. Our simulation tool provides support by collecting sensory data to be used in the development of novel perception algorithms for GRIFFIN robots. We believe this sensing simulation scheme can contribute and boost R&D in perception systems for ornithopter robots.

Summarizing, the contribution of this work is two-fold. First, it presents a simulation tool to generate simulated multi-sensor measurements obtained from the sensors onboard an ornithopter during landing and perching maneuvers. The simulated sensors include those with the highest interest in ornithopters perception such as frame-based cameras, event-based sensors, solid-state LiDARS, range sensors, altimeters, among others. Second, a dataset with the sensor measurements provided by our simulator is published online. The recorded sensor information corresponds to the simulation of several landing and perching trajectories in different scenarios. This paper is structured as follows. Section 2 describes the related work of bioinspired trajectory generation and mobile robotic simulation. The design methodology of simulation is presented in Section 3. Section 4 defines the simulation architecture explaining each block of the simulator. The dataset generation description is explained in detail in Section 5. Finally, Section 6 includes the conclusions and future work.

2 Related work

Ornithopter simulators mainly focus on the study of dynamics and flight control. For instance, the stability and controllability of the aircraft are analyzed in [6] to control the non-linear flight of a flapping-wing robot. The flexible multibody dynamics of an ornithopter is simulated in [7] considering fluid-structure interaction and flight dynamic behavior to design a robotic model capable of flying in trim. The flappy hummingbird [8] is an open source dynamic simulator of flapping-wing micro aerial vehicles created to facilitate the design and validation of flight control architectures for flapping-wing robots. Although these tools are useful to simulate ornithopters dynamics, kinematics and aerodynamic effects, to the best the author's knowledge there is not a simulation tool designed to retrieve sensor information during their flight.

Robotic simulation tools are useful for testing and evaluating robotics algorithms before applying them on the real platforms. The use of simulators enlarge security and optimize time by evaluating the behavior of robots in challenging scenarios without endangering the platform and users. Among the most popular robotic simulators, there are Gazebo [9] and V-REP [10]. Game engines are also used for the development of robotic simulators. Engines are preferred in applications that require high framerate and photorealistic rendering. Airsim [11] and CARLA [12] are some of the examples of game engine simulators used for robotics and artificial intelligence applications. Our multi-sensor simulation architecture retrieves the sensor measurements during the flight of ornithopter robots by integrating two simulation tools in parallel; Gazebo and Unreal Engine 4 (UE4). The former simulates different sensors such as frame-based cameras, lasers, force and range sensors. The last is part of the event camera simulator [13], the benefit of using this sensor in our application is explained in Section 3.2. Both simulators are integrated in the Robot Operating System (ROS).

3 Design

The aim of this work focuses on retrieving simulated multi-sensor measurements for robotics perception during the landing of an ornithopter. For this purpose, we propose the design of a tool to simulate different perception sensors that includes a bioinspired trajectory generator to retrieve the type of movements exerted by birds for landing and perching. Thus, simulation development has to satisfy the following design requirements. First, the platform has to be capable of simulating bioinspired trajectories to resemble the movements performed by birds during flight. Tau theory describes the principle used by animals and humans to guide their motion to make contact with an object or surface using the time-to-contact as reference. The theory has been used in aerial robotics to guide and control the motion of multirotor platforms for docking and landing [14].

Second, the simulator has to include the type of sensors most widely used for robotics perception. The hardware sensor selection takes into consideration the type of information necessary for localization, mapping, obstacle avoidance, and object identification. We analyze the problem from a robotic perception perspective and select the sensors required for these tasks. However, resembling the flight of birds involves continuous fast movements and strong scene changes in dynamic environments. These facts constrain the hardware selection: the sensors should process information and correct their measurements with the lowest possible noise at the highest frame rate. Our sensor selection considers a variety of sensors typically used in aerial robotics such as LiDAR, IMU, altimeters and vision sensors. It also integrates event-based cameras to deal with the fast-motion limitations and low-latency measurements with high dynamic range. Fusing events from event cameras together with classical perception information (e.g images, point clouds, and IMU measurements) increases the sensor robustness for the development of future SLAM and object detection algorithms for flapping-wing platforms. The most relevant aspects about the planning of the *tau trajectory* and a brief introduction to event cameras functioningand applications are detailed below.

3.1 Tau theory planning

The simulation architecture presented in this paper relies on *tau theory* [15] to approximate the landing and perching maneuvers of ornithopters. Tau theory postulates that humans and animals (i.e. notably birds) use a combination of simple strategies and the value of *tau* to guide and control most of their intended movements. Tau (τ) is a variable related to the time an observer would take to contact an object or surface if the speed remains constant. The value of τ provides a first order approximation of the time-to-contact (TTC) for a given gap χ:

$$\tau(t) = \frac{\chi(t)}{\dot{\chi}(t)}, \tag{1}$$

where $\chi(t)$ and $\dot{\chi}(t)$ are the gap and its rate of closure at time step t. The gap χ is, by convention, always negative and the initial closure rate $\dot{\chi}(0)$ is positive. The work in [15] found that birds tend to keep the gap closure rate constant to control their deceleration. This behavior was defined as a constant tau-dot strategy. Subsequent research showed that zero velocity at contact is reached when $\dot{\tau}$ is kept constant, positive and less than 0.5. Another variant of tau-dot strategy was proposed in [14] to guide a breaking maneuver using $\hat{\tau}(t) = kt + \tau(0)$, which entails a more practical approximation to tau-dot strategy, i.e. the computation of $\dot{\tau}$ is avoided. It is worth noting that the tau theory postulates that animals do not require cognitive processing for TTC since it is available at neural circuit level [15]. Thus, the value of a gap and its closure rate can be computed as:

$$\chi(t) = \chi(0)\Big(1 + kt\frac{\dot{\chi}(0)}{\chi(0)}\Big)^{\frac{1}{k}}$$

$$\begin{aligned}
\dot{\chi}(t) &= \chi(0)\Big(1 + kt\frac{\dot{\chi}(0)}{\chi(0)}\Big)^{(\frac{1-k}{k})}, \\
\ddot{\chi}(t) &= \frac{\dot{\chi}(0)^2}{\chi(0)}(1-k)\Big(1 + kt\frac{\dot{\chi}(0)}{\chi(0)}\Big)^{(\frac{1-2k}{k})}
\end{aligned} \tag{2}$$

where $\chi(0)$ is the initial gap value, and $0 < k \leq 0.5$ guarantees the gap and its closure rate reach zero at the same finite time $T = \frac{-\tau(0)}{k}$. A further analysis shows that for $0.5 < k < 1$ the gap closes to zero with $\dot{\chi}(0) \neq 0$ leading to a collision. Figure 1 shows the gap closure for different values of k. It is worth mentioning that the figures of this paper show the main gap (e.g z) positive contrary to the convention being consistent with a landing trajectory.

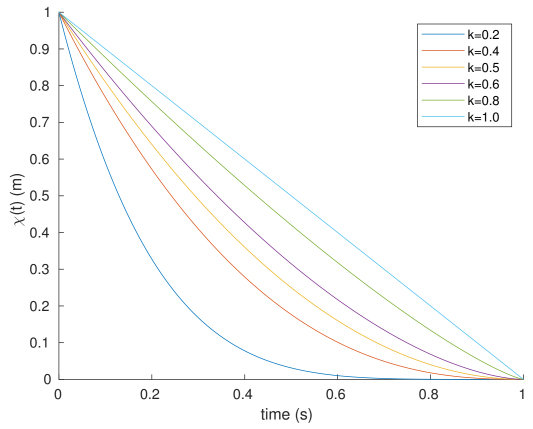


Fig. 1: Trajectories χ for $0 < k \leq 1$ closing the gap at $t = 0$. For $k = 1$ the gap closes with $\dot{\chi}$ constant producing a collision.

The simulation of landing or perching trajectories of a bird requires the closure of gaps for position and orientation. Therefore, an extension of tau-constant strategy for more than one gap is required. Closing multiple gaps can be achieved through tau-coupling strategy, which consists of coupling additional gaps ϑ to the main gap χ. The tau-coupling is computed as $\tau^{\chi} = \kappa\tau^{\vartheta}$, where κ is a closure constant ratio between both gaps. Therefore, a coupled gap ϑ, its closure rate $\dot{\vartheta}$, and acceleration $\ddot{\vartheta}$ at time t are obtained as:

$$\begin{aligned}
\vartheta(t) &= c\chi(t)^{\frac{1}{\kappa}-1}, \\
\dot{\vartheta}(t) &= c\frac{1}{\kappa}\dot{\chi}(t)\chi(t)^{\frac{1}{\kappa}-1}, \\
\ddot{\vartheta}(t) &= c\frac{1}{\kappa}\bigg(\Big(\frac{1}{k}-1\Big)\dot{\chi}(t)^2 + \chi(t) + \ddot{\chi}(t)\bigg)\chi(t)^{\frac{1}{\kappa}-2},
\end{aligned} \tag{3}$$

where $c = \frac{\vartheta(0)}{\chi(0)^{(1/\kappa)}}$.

In the context of the proposed *tau trajectory* generation, the trajectory is obtained by closing a main gap in the z-axis and three coupled gaps in the x and y axis, and the roll θ, pitch ϕ and yaw ψ angles. For simplicity, each gap will be denoted with their corresponding superscript, and a subscript will denote the moment of time. For instance, the main gap in the z-axis and a couple gap on the yaw ψ at time t will be denoted χ_t^z and ϑ_t^ψ respectively. The trajectory S_t at time t includes the gap values, their closure velocities, and accelerations represented in a reference frame F aligned with the goal position and orientation. The transformation of F, the reference frame to a global reference frame, can be computed in a straightforward manner. Hence, the proposed method is applicable to any set of initial and target configurations. The proposed method for tau-theory based trajectory generation provides a compact way of estimating trajectories that are similar to those that birds perform during landing and perching maneuvers.

3.2 Event Cameras

Event cameras are silicon retina vision sensors that mimic the neural behavior and architecture of the retina. These devices are modifying the paradigm of retrieving scene information using vision sensors. Unlike frame-based traditional cameras, event-driven sensors provide information based on pixel intensity variation. Events are triggered asynchronously whenever the intensity of a pixel exceeds a specific threshold and transmitted using the Address Event Representation (AER) Protocol. The sensor provides high temporal resolution generating events with a resolution of μ seconds. Besides, event cameras offer a high dynamic range (140 dB) and low power consumption. These capabilities have increased the interest of the computer vision and robotic communities to use these sensors in applications with challenging conditions for frame-based cameras.

Events are defined as a tuple of the form $e = (t, x, y, p)$, where t is the time in which the events are triggered, (x, y) is the pixel position, and p is the polarity (i.e. either 1 or 0). The event data format limits the use of events in computer vision and robotics applications as most of the state of the art algorithms cannot be directly applied to the stream of events. During the last years, different novel methodologies have been developed to adapt classical algorithms to use events in applications such as feature detection and tracking [16], optical flow estimation [17], visual odometry [18] and SLAM [19]. These advances contribute to the research in a novel area and lead the sensor integration between event-driven sensors and other sensors such an IMU and frame-based cameras [20].

Few event camera simulators have been reported. In [21] the difference between consecutive images is used to generate edges that resemble the events produced by the edges of moving objects. The method proposed in [22] attempts to simulate the behavior of an event camera by using the high frame rate capabilities of Blender to sample images like in a continuous timeline. This feature adds the low-latency properties of event cameras. The simulator was improved in [13] by simulating the asynchronous behavior of the retinas by triggering events based on the prediction of the image dynamics.

4 The Architecture of the Presented Simulator

The presented multi-sensor simulation tool adopts an architecture based on ROS integrating Gazebo and *Unreal Engine* 4 (UE4). It combines the advantages of ROS to integrate drivers, packages, and state of the art robotics algorithms with the advantages of UE4 to provide fast realistic rendering, easy portability, and an intuitive block programming interface. ROS handles the bioinspired trajectory generator based on *tau theory* to simulate the trajectories performed by birds during landing and perching. It also integrates different sensor simulators such as feature-based cameras, event cameras, and proximity sensors; and publishes the sensor measurements on ROS topics. On the other hand, UE4 handles the simulation of a photo-realistic version of the scenario to capture images of the scene at a very high frame rate. Gazebo simulates a simple version of the environment for sensors such as the Velodyne, lasers, sonar, and other sensors.

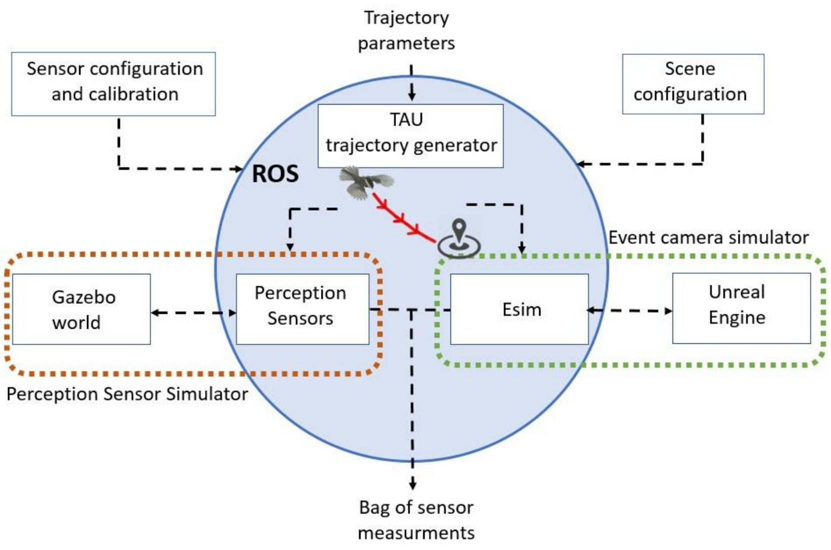


Fig. 2: Block diagram of the simulation architecture.

Figure 2 shows the block diagram of the architecture. The system receives three types of inputs. First, the parameters to compute the *tau trajectory* S_t as described in Eqs. (2) and (3). Second, the scene configuration file with the pose and orientation of each object in the scene. Third, the sensors configuration and calibration. For instance, the simulation model of the frame-based camera requires the camera calibration file in *yaml* format including the camera matrix and the optical distortion coefficients. Each simulated sensor requires the definition of the coordinate frame of the sensor w.r.t. the reference frame of the simulated robot. The proposed simulation tool includes the following sensors: event camera, Velodyne HDL-32, frame-based camera, laser, altimeter, and

IMU. Due to its flexible and modular architecture, it can be easily extended by including additional devices such as stereo cameras and depth sensors. The output of the simulation tool is a rosbag file with the measurements taken by the simulated sensors while following trajectory S_t also included in the recording file.

The simulation tool is divided into three main modules: the *Tau Trajectory Generator*, the *Perception Sensors Simulator* and the *Event Camera Simulator*. The *Tau Trajectory Generator* module computes trajectory S_t using the approach described in Section 3.1. The proposed implementation generates up to the 18 components to define the pose $(\vartheta_t^x, \vartheta_t^y, \chi_t^z, \vartheta_t^\theta, \vartheta_t^\phi, \vartheta_t^\psi)$, the velocities $(\dot{\vartheta}_t^x, \dot{\vartheta}_t^y, \dot{\chi}_t^z, \dot{\vartheta}_t^\theta, \dot{\vartheta}_t^\phi, \dot{\vartheta}_t^\psi)$, and accelerations $(\ddot{\vartheta}_t^x, \ddot{\vartheta}_t^y, \ddot{\chi}_t^z, \ddot{\vartheta}_t^\theta, \ddot{\vartheta}_t^\phi, \ddot{\vartheta}_t^\psi)$ for each time t. The inputs of the trajectory generation block are the initial pose in the target reference frame F, and the parameters k^z,κ^x, κ^y, κ^θ, κ^ϕ, and κ^ψ , where the upper script defines the tau theory gap.

The *Perception Sensors Simulator* module contains the simulation model of sensors such as Velodyne, altimeter, and IMU. The simulator moves the reference frame of the sensors in the virtual scenario by following the *tau trajectory*. The input of the simulation tool includes the external calibration (transformation matrix of the sensor local reference frame w.r.t. the ornithopter frame) and internal calibration files for each simulated sensor.

The *Event Camera Simulator* is based on *esim* [13]. The simulator was modified to receive the input from our *tau trajectory* generator keeping the correct orientation during the simulation. Further, the reference frame of the sensor is modified by a transformation matrix that references it to the reference frame of the scene. The simulator runs in ROS and uses Unreal Engine 4 to render images from a 3D scenario at a very high frame rate (from 100 Hz to 1 kHz). The asynchronous behavior of event cameras is achieved in simulation using two key facts: (i) rendering images at very high frame rate on the engine, and (ii) sampling frames adaptively to generate events asynchronously using the prediction of the optical flow. The simulator moves the event camera following the sensor external calibration and S_t, the trajectory computed using *tau theory*.

5 Datasets

The dataset provides the simulated sensor measurements obtained during the landing of the ornithopter using different landing trajectories. For each simulation, the onboard sensors move by following the trajectory obtained with the tau trajectory generator. Each trajectory was computed by using a final approach angle of $\pi/6$ and an initial velocity in a range between 3 to 5 m/s as in a real ornithopter. The trajectories were sampled at $\Delta t = 0.1$s until $t = T$, where T is computed as described in Section 3.1. Figure 3 shows different normalized trajectories obtained from the tau trajectory generator when performing the simulated landing maneuver. The trajectories were computed by varying the values of κ^x and κ^y as it is described in the figure, and setting $k^z = 0.5$ to obtain a smooth descending without colliding with the ground. The robot poses along the trajectories are shown as vectors. The position of the vector represents the ornithopter position in (x, y, z) while its magnitude and direction are defined by the linear velocity components at each point of the trajectory.

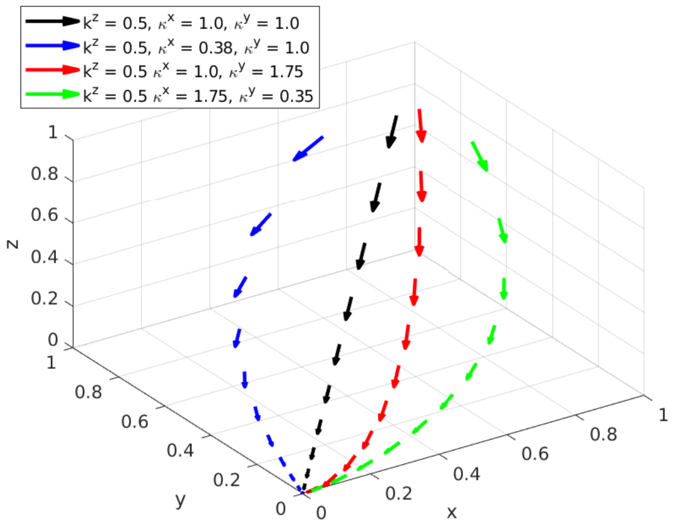


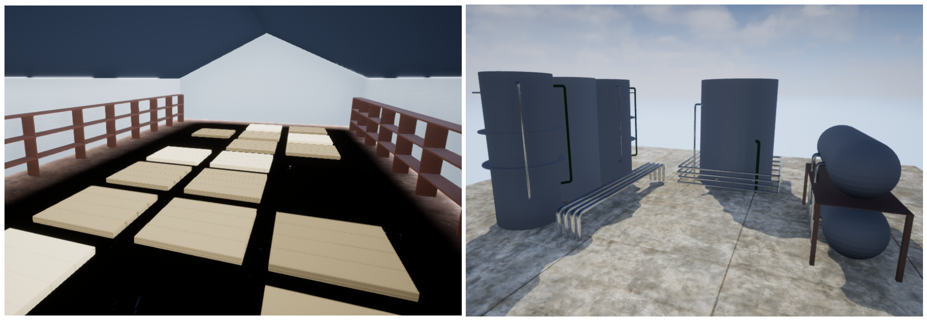
Fig. 3: Different normalized trajectories describing the simulated landing maneuver of the ornithopter. The figure shows the magnitude and direction of the approach velocity vectors.

Several simulation scenarios were developed. The scenarios were inspired by industrial environments such as warehouses and factories (Fig. 4) where ornithopter robots could provide robotic support on tasks such as surveillance, payload delivery, and remote sensing. The objects in the developed scenarios were designed in *Blender*. Additional scenarios can be integrated by simply importing the models of the scenes in both Gazebo and UE4. The model integration preserves the scene configuration (i.e position, orientation, and scale) to avoid mismatches between the sensor measurements from each simulator.

In general, the time to simulate the sensor measurements t_s along the trajectory is longer than the trajectory duration as the simulator performs several computations to produce all the multi-sensory information. However, it is worth to mention that the reference time of the measurements does not correspond to t_s as it is referred to the trajectory timestamp t_t. The publish rate of the dataset can be adjusted while running the bagfile by setting the factor r of ROS to t_s/t_t.

Our dataset is divided into six simulations each for a different landing trajectory. For each trajectory, the measurements from the monocular camera, event camera, sonar, IMU and Velodyne LiDAR are recorded in a rosbag file. The dataset also includes the ground truth pose information of the sensors during the landing maneuver, see Fig. 5. Our architecture allows the integration of additional sensors available on the *gazebo_ros_package* such as Kinect, lasers and depth cameras. Each simulation of the dataset contains the following files:

- The rosbag file with the sensor measurements.
- A file with the instructions to run the bag.
- The input bioinspired trajectory computed using *tau theory*.
- The events generated during the simulation in a text file using the format (timestamp, x, y, polarity).
- The model of each object of the scene including a file with the object poses.



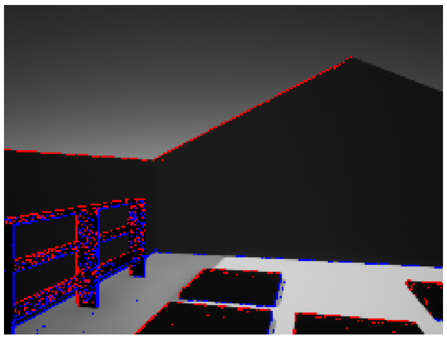
(a) (b)

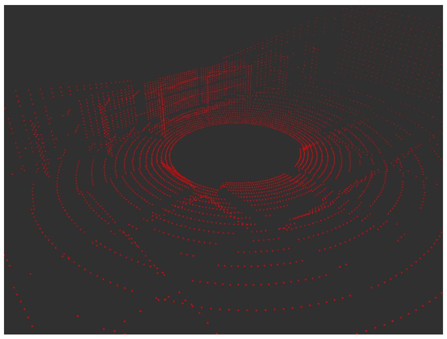
Fig. 4: Simulation scenarios to perform the sensors measurements simulation. A warehouse (a), and an oil refinery (b).

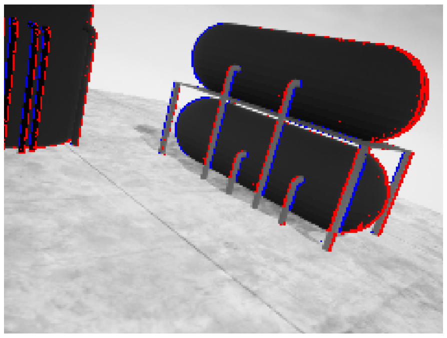
6 Conclusions

In this work, we introduce a simulation tool to retrieve the sensing information captured during the landing and perching maneuvers of ornithopter robots. The operation of ornithopters poses very strong perception challenges, which contrasts with the effort-demanding implementation and testing with these aerial platforms. This work is motivated by the need to retrieve the missing realistic multi-sensory measurements to develop, debug and tune perception and autonomous navigation methods for ornithopter robots. Our simulator moves the sensors describing bioinspired trajectories computed using *tau theory*.

A dataset with several trajectories and sensor measurements is publicly available online. Each dataset provides the bioinspired trajectory followed during the simulation along with the sensing information from different sensors. We hope that our dataset boosts the development of novel perception algorithms for robot localization, mapping, and object detection for ornithopters before the development of a platform capable of carrying the necessary sensors for robotic perception. Finally, our future work points towards the development of such perception


(a)


(b)


(c)

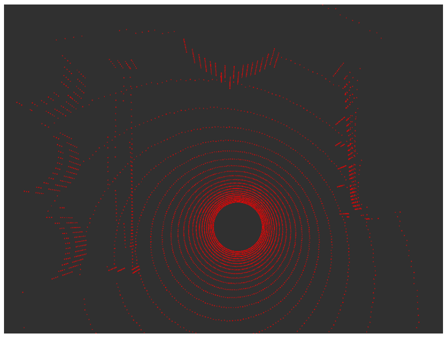

(d)

Fig. 5: Example of the sensor measurements obtained during the simulation of a landing trajectory on the warehouse and refinery scenarios. Rendered images including the generated events during a time window of 10 ms (a,c), Point cloud from the simulated Velodyne LiDAR (b,d). The lighting of image (a) was modified for better visualization.

algorithms particularly aiming to fuse the information provided by event cameras and classical perception sensors to solve the perception challenges presented by the flight of flapping-wing vehicles.

Acknowledgements: This work was supported by the European Research Council as part of GRIFFIN ERC Advanced Grant 2017 (Action 788247) and the ARM-EXTEND project funded by the Spanish National RD plan (DPI2017-89790-R).

References

1. G. Croon, M. Perçin, B. Remes, R Ruijsink, and C. Wagter. *The DelFly*. 2016.
2. G. Folkertsma, W. Straatman, N. Nijenhuis, C. Venner, and S. Stramigioli. Robird: a robotic bird of prey. *IEEE Robotics & Automation Magazine*, 24(3):22–29, 2017.

3. Festo Corporate. Smartbird. https://www.festo.com/net/SupportPortal/Files/46270/Brosch_SmartBird_en_8s_RZ_110311_lo.pdf, 2011.
4. Festo Corporate. Bionicflyingfox. https://www.festo.com/net/SupportPortal/Files/492827/Festo_BionicFlyingFox_en.pdf, 2018.
5. D. Lee, R. Bootsma, M.e Land, D. Regan, and R. Gray. Lee's 1976 paper. *Perception*, 38(6):837–858, 2009.
6. J. Han, J.and Lee and D. Kim. Ornithopter modeling for flight simulation. In *2008 Int. Conf. on Control, Automation and Systems*, pages 1773–1777, 2008.
7. A.T Pfeiffer, J. Lee, J. Han, and H. Baier. Ornithopter flight simulation based on flexible multi-body dynamics. *Journal of Bionic Engineering*, 7(1):102–111, 2010.
8. F. Fei, Z. Tu, Y. Yang, J. Zhang, and X. Deng. Flappy hummingbird: An open source dynamic simulation of flapping wing robots and animals. *arXiv preprint arXiv:1902.09628*, 2019.
9. N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ IROS 2004*, volume 3, pages 2149–2154, 2004.
10. E. Rohmer, Surya PN Singh, and M. Freese. V-rep: A versatile and scalable robot simulation framework. In *IEEE/RSJ IROS 2013*, pages 1321–1326, 2013.
11. S. Shah, C. Dey, D.and Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, pages 621–635, 2018.
12. A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. In *Proc. of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.
13. H. Rebecq, D. Gehrig, and D. Scaramuzza. Esim: an open event camera simulator. In *Conference on Robot Learning*, pages 969–982, 2018.
14. F. Kendoul. Four-dimensional guidance and control of movement using time-to-contact: Application to automated docking and landing of unmanned rotorcraft systems. *The Int. Journal of Robotics Research*, 33(2):237–267, 2014.
15. D. Lee. General tau theory: evolution to date. *Perception*, 38(6):837, 2009.
16. V. Vasco, A. Glover, and C. Bartolozzi. Fast event-based harris corner detection exploiting the advantages of event-driven cameras. In *IEEE/RSJ IROS 2016*, pages 4144–4149, 2016.
17. R. Benosman, C. Clercq, X. Lagorce, S. Ieng, and C. Bartolozzi. Event-based visual flow. *IEEE Transactions on Neural Networks and Learning Systems*, 25(2):407–417, 2014.
18. B. Kueng, E. Mueggler, G. Gallego, and D. Scaramuzza. Low-latency visual odometry using event-based feature tracks. In *IEEE/RSJ IROS 2016*, pages 16–23, 2016.
19. H. Rebecq, T. Horstschäfer, G. Gallego, and D. Scaramuzza. Evo: A geometric approach to event-based 6-dof parallel tracking and mapping in real time. *IEEE Robotics and Automation Letters*, 2(2):593–600, 2017.
20. A. 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 Robotics and Automation Letters*, 3(2):994–1001, 2018.
21. J. Kaiser, J. Tieck, C. Hubschneider, P. Wolf, M. Weber, M. Hoff, A. Friedrich, K. Wojtasik, A. Roennau, R. Kohlhaas, et al. Towards a framework for end-to-end control of a simulated vehicle with spiking neural networks. In *2016 IEEE Int. Conf. SIMPAR*, pages 127–134, 2016.
22. E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM. *The Int. Journal of Robotics Research*, 36(2):142–149, 2017.