IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2021.3066529

Auto-tuned event-based perception scheme for intrusion monitoring with UAS

JUAN PABLO RODRíGUEZ-GÓMEZ[1], AUGUSTO GÓMEZ EGUíLUZ[1], JOSÉ RAMIRO MARTíNEZ-DE DIOS[1], ANIBAL OLLERO[1], (Fellow, IEEE)

[1]GRVC Robotics laboratory, University of Seville, Seville 41092, Spain (e-mail:{jrodriguezg, ageguiluz, jdedios, aollero}@us.es)

Corresponding author: Juan Pablo Rodríguez-Gómez (e-mail: jrodriguezg@us.es).

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 (Grant H2020-2019-871479). Partial funding has been received from project ARM-EXTEND funded by the Spanish RETOS R&D Program by the Ministry of Science and Innovation under Grant DPI2017-89790-R.

ABSTRACT This paper presents an asynchronous event-based scheme for automatic intrusion monitoring using Unmanned Aerial Systems (UAS). Event cameras are neuromorphic sensors that capture the illumination changes in the camera pixels with high temporal resolution and dynamic range. In contrast to conventional frame-based cameras, they are naturally robust against motion blur and lighting conditions, which make them ideal for outdoor aerial robot applications. The presented scheme includes two main perception components. First, an asynchronous event-based processing system efficiently detects intrusions by combining several asynchronous event-based algorithms that exploit the advantages of the sequential nature of the event stream. The second is an off-line training mechanism that adjusts the parameters of the event-based algorithms to a particular surveillance scenario and mission. The proposed perception system was implemented in ROS for on-line execution on board UAS, integrated in an autonomous aerial robot architecture, and extensively validated in challenging scenarios with a wide variety of lighting conditions, including day and night experiments in pitch dark conditions.

INDEX TERMS Event-based vision, intrusion detection, surveillance, UAV.

I. INTRODUCTION

UNMANNED Aerial Systems (UAS) have attracted high interest in large-area surveillance and monitoring applications. Although visual cameras are the most commonly adopted sensors for automatic vision-based surveillance and intrusion detection using UAS, they face relevant problems in large, complex, and unstructured scenarios. Robustness to lighting conditions is a critical issue in automatic vision systems. It is often addressed by combining images from different onboard cameras (e.g., visual and infrared cameras), which affects the UAS payload, electrical consumption, and computational needs, reducing the UAS flight time. Motion blur is also a severe problem in highly dynamic or poorly-illuminated scenarios.

This paper uses event cameras for UAS-based surveillance. Event cameras capture visual information in the form of events representing changes of intensity in the camera pixels, which are triggered asynchronously with a high temporal resolution (order of µs). Event cameras have very wide dynamic range, suitable for robust operation under a wide variety of lighting conditions. They are insensitive to motion blur, and have low power consumption. Several commercial event camera models can also provide visual images, enabling combined event-visual processing, see e.g., [1], [2], [3], or [4], among others. A good number of successful event-based techniques have been proposed in the last years, [5]. Most of them group the received events in frames called event images. Event images enable designing elaborated frame-based processing schemes, but they cannot always fully exploit the sequential and asynchronous capabilities of the event cameras. Besides, event images could lead to an overestimation of the scene representation, a similar phenomenon to motion blur in traditional images [6].

Similarly to visual-based processing, event-based techniques require adapting its parameters to the conditions of the addressed problem. Most existing event-based techniques adopt empirical or manual parameter selection, which often leads to inefficient long trial-and-error iterative processes. To

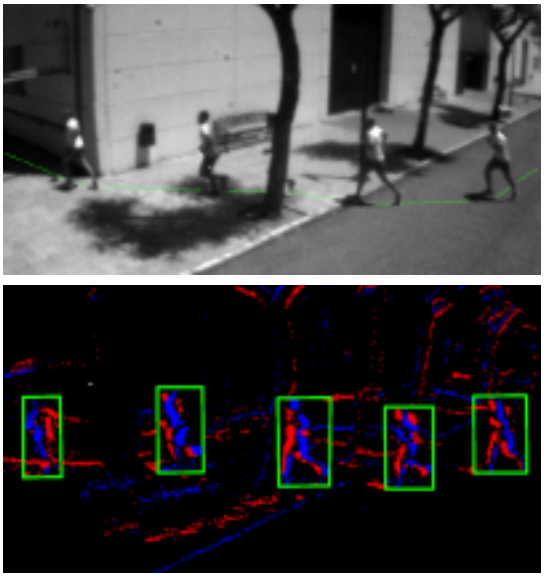

FIGURE 1: Results of the proposed event-based intrusion monitoring system in UAS-based experiments: visual images (left) and events (right).

the best of the author's knowledge no technique to automatize the selection/tuning of the parameters of event-based vision system has still been reported.

This paper presents an asynchronous event-based processing scheme for intrusion monitoring with UAS. It includes two main perception components. First, an asynchronous event-based processing system detects intrusions on-line using only the events. It performs event-by-event processing in all the algorithms involved resulting in a fully asynchronous processing system that exploits the advantages of event cameras. It was carefully designed and endowed with mechanisms to reduce the computational cost in order to enable on-line onboard execution. The second block is an off-line training mechanism that adjusts the event-processing parameters to a particular surveillance scenario and mission exploiting the combined processing of the event stream and visual images, both provided by a DAVIS346 event camera. The proposed scheme and techniques have been implemented in ROS, integrated within an autonomous UAS waypoint-based navigation system, and validated in UAS surveillance missions in challenging scenarios, see Figure 1.

This paper is inspired by some ideas sketched in [7]. The main improvements over [7] are: (i) a new auto-tuning system for adapting the parameters of the intrusion monitoring scheme to a particular surveillance scenario; (ii) robustness improvement in the event-based corner detection, tracking, and clustering algorithms of the intrusion monitoring scheme; (iii) development, integration, and validation of the proposed scheme in an aerial robot architecture for autonomous surveillance; and (iv) new experimental validation in challenging scenarios and robustness evaluation against lighting conditions.

The rest of the paper is organized as follows. Section II briefly summarizes the main works directly related to the addressed topics. The proposed scheme is presented in Section III. The asynchronous event-based processing system is described in Section IV. The functionality for adapting the event-based processing to the particular surveillance scenario and mission is summarized in Section V. Section VI presents the experimental validation and robustness analyses. Section VII concludes the paper and highlights future research steps.

II. STATE OF THE ART

Automatic UAS-based surveillance and intrusion monitoring using visual sensors are intensely-researched topics where many methods and systems have been developed focusing on perception, planning, or multi-robot coordination, among others. Many of them adopt visual cameras as main sensors and suffer from the limitations of traditional cameras, such as lighting conditions and motion blur, which significantly constrain their applicability. To address the sensitivity to lighting conditions, other methods use combinations of several cameras, such as visual and infrared cameras. Besides increasing the payload, energy consumption, and computational power, some of these additional cameras (e.g., infrared cameras) are particularly sensitive to motion blur, which hamper their effectiveness in dynamic environments.

Event cameras have attracted increasing interest in the robotics and computer vision communities [5]. Recently, event cameras have been proposed for intrusion monitoring and detection of humans. The work in [8] detected pedestrians with a static DAVIS camera by fusing the confidence maps of two YOLO V3 classifiers, one for image frames and, the other, for event images. Their method improved processing rate and detection precision over using only visual images. Two methods for face detection using event cameras were presented in [9]. The authors compared the performance when using the image frames from a visual camera, image frames reconstructed from events, and event images captured every 20 ms. Although the best precision was obtained with the visual camera, the results show the viability of using only events for face detection.

Using event cameras on board robots requires the development of methods to deal with the additional event generation due to the camera motion. In [10], camera rotation was estimated through a contrast maximization process using as reference event images with polarity. A clustering-based method was proposed in [11] to produce event-compensated images by estimating the motion parameters of each cluster using an alignment maximization approach. Work [12] used neural networks for independent motion detection by estimating camera ego-motion. Recently, the work in [13] aligned curves described by event trajectories with time-varying motion parameters to perform motion compensation. However, none of the above methods were evaluated on board mobile robots and, therefore, they did not deal with the issues caused by the vibrations and movement of the robot. Event cameras have been recently employed on board

UAS for different perception problems. A model of the affine transformation between two consecutive event images was used in [14] to compensate for the global motion of a Micro Aerial Vehicle (MAV), and thus, the resulting events were assumed to represent the moving objects. The method described in [6] included both downward-facing and front-facing event cameras to perform obstacle avoidance on board UAS. The events were accumulated in event images for obstacle detection using shallow deep neural networks for event noise removal, homography computation, and segmentation through flow estimation. Then, a controller guided the UAS towards a safe direction opposite to the motion of the incoming obstacle. Recently, the work in [15] employed batches of events collected each 10 ms to perform ego-motion compensation for obstacle detection using a stereo event camera set-up mounted in a quadrotor. The UAS performed a reactive avoidance strategy based on potential fields describing obstacles as geometric primitives. Additional progress has been done towards the use of event cameras for UAS control. In [16] an autonomous MAV landing approach based on the optical flow of event images obtained from a downwards-pointing DVS sensor was presented exhibiting high accuracy at high landing speeds. Further, the work in [17] exploited the μs resolution of event cameras to control the attitude of a dual-copter platform.

All the aforementioned works process event images generated by accumulating events. In general, event images enable designing elaborated processing schemes similar to those from traditional computer vision, although on the other hand, sacrifice some of the advantages of event camera μs resolution. In fact, some works, e.g., [6], include extra mechanisms for reducing motion blur originated in event images. Due to its higher computational demands, asynchronous *event-by-event* processing has been mainly used for low-level processing techniques. The work in [18] proposed a Harris-inspired asynchronous corner detection based on events. The authors in [19] presented an asynchronous method to detect corners by checking the values of a Surface of Active Events (SAE) [20] for the coordinates corresponding to two circles around the last event coordinate. A variation of this method was proposed [21] to consider arcs greater than 180°, enhance detection speed, and track the corners using graph trees. Their tracking method was extended in [22] to deal with jittery effects from multiple corner detection. A mean shift method adaptation for event clustering was presented in [23]. Additionally, the authors proposed a cluster tracker using Bayesian filtering. Despite their proposed approach being robust to different cluster shapes and velocities, it was only suitable for static backgrounds (i.e., fixed camera). An asynchronous visual inertial odometry solution was presented in [24]. Although asynchronous event processing methods provide reliable solutions to localization, feature detection, tracking, and clustering, their employment in robots navigating in realistic, complex, and unstructured scenarios is still an under-researched area. In a recent work [25], a hybrid approach (i.e., event-by-event and event images) was developed for asynchronous line tracking that was used in a visual servoing scheme to perform time-to-contact maneuvers using a multirotor aerial platform.

The performance of some of the previous event-based methods, such as [18], [21], and [15], depend on fine-tuning their algorithm parameters. Meta-heuristic optimization strategies such as Simulated Annealing (SA) have been used for parameter tuning in computer vision and robotic applications, such as image segmentation [26], motion blur removal [27], feature selection [28], and robot path planning [29]. However, the parameter tuning of event-based vision algorithms is typically performed empirically. In this work, SA is used in a semi-supervised training system for off-line tuning the parameters of the event-based processing algorithms. To the best of the author's knowledge, it is the first fine-tuning system for an event-based vision method.

This paper presents an asynchronous event-based processing scheme for UAS-based intrusion monitoring. Its main contributions are:

- a fully asynchronous event-based intrusion detection system for on-line execution on board a UAS;
- an off-line training system for fine-tuning the event-processing parameters to a particular surveillance scenario and mission;
- the proposed scheme has been integrated within an autonomous UAS navigation architecture and validated in realistic challenging scenarios.

III. INTRUSION MONITORING SYSTEM FOR UAS

The objective of this work is to develop a full scheme for autonomous surveillance and intruder monitoring in realistic, complex, and unstructured outdoor scenarios using UAS equipped with event cameras. The UAS performs periodic or on-demand surveillance tours. The scenario can change from one tour to another, but in the same tour it is assumed mostly static. This is the case in many security and inspection applications, e.g., industry, building perimeter, or frontier surveillance, among others. The intrusion monitoring scheme should be robust to lighting conditions and be operative during day and night. Besides, it should be robust to motion blur effects, which can be particularly severe in dark lighting conditions. Also, it should be easily particularized and adapted to the conditions of the specific scenario.

The proposed scheme relies on event cameras, which provide high dynamic range and temporal resolution –hence, high robustness against lighting conditions and motion blur. They have low power consumption and are small. Hence, they are suitable for integration on board small UAS. Besides, their high dynamic range make them suitable for day/night operation, enabling the use of only one camera instead of configurations with one camera for day vision and another for night vision. Also, their insensitivity to motion blur make event cameras robust to the potential mechanical vibrations occurring during UAS flight.

The main functional modules of the proposed scheme are summarized in Figure 2. The *Navigation* system includes

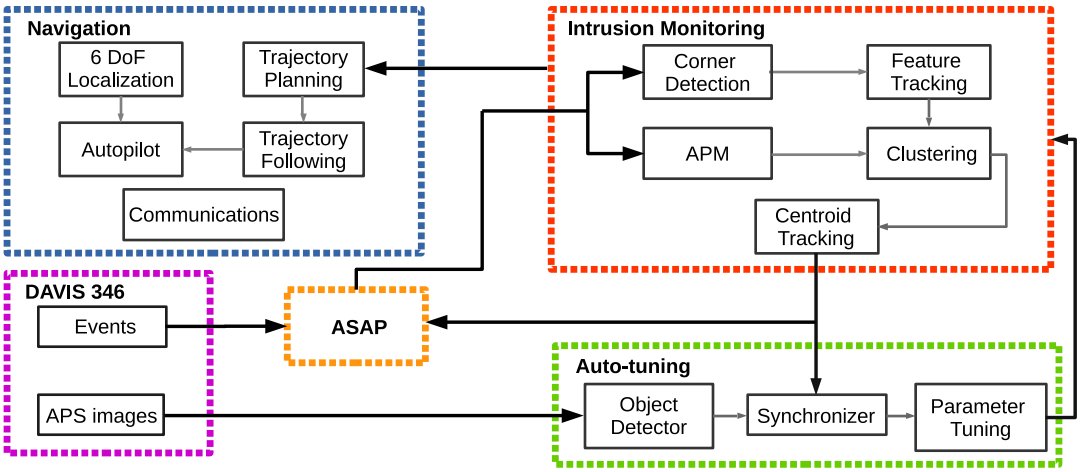

FIGURE 2: Main functional modules of the proposed asynchronous event-based scheme.

modules for trajectory planning, waypoint following, and 6-DoF localization. The trajectory planning was performed using the Lazy Theta* planner described in [30]. 6-DOF localization was performed fusing RTK GPS and IMU measurements. This navigation system is not a main contribution of the paper, and for brevity, its detailed description is omitted. The UAS is assumed to be equipped with a DAVIS camera, which includes an event-based dynamic vision sensor (DVS) and an image-based active pixel sensor (APS). During the surveillance mission execution, the events from the DVS sensor are processed on-line and on board by the *Intrusion Monitoring* (*IM*), which implements an asynchronous event-based processing scheme for intrusion detection. The events generated by the DVS sensor are packaged using *ASAP* [31], which synchronizes event packaging and event processing by adjusting the number of events sent for processing such that the provided events are processed as soon as possible while avoiding processing overflow. During the training stage, the parameters of *IM* are off-line adapted for a given surveillance scenario by the *Auto-tuning* system, which implements an optimization method based on Simulated Annealing. The method maximizes the similarity between the results provided by *IM* and a ground truth reference obtained by an automatic object detector (YOLO V3 [32] in the reported experiments) fed with the visual images from the DAVIS APS sensor. Of course, the proposed scheme could be extended to event cameras with no APS sensor by using an additional onboard visual camera.

The proposed system operates similarly to a "robotic security guard"'. The UAS, periodically or on demand, autonomously executes surveillance tours performing on-line intrusion monitoring, reporting, and logging. The robot trajectories are defined by sequences of waypoints selected for their good visibility of the scenario. When the robot reaches one waypoint it stays in steady flight (hovering) while performing intrusion monitoring using *IM*. Even during hovering, the static scenario background originates events due to residual UAS motions and vibrations, requiring specific mechanisms to distinguish between events created by moving targets and those created by the static background. If no intrusion is detected at that waypoint, the robot keeps its trajectory to the next waypoint. If an intrusion is detected, it is reported to the *Ground Station*. Depending on the surveillance policies adopted, the robot can continue the surveillance tour to avoid compromising the rest of the scenario or, it can keep monitoring the zone with the detected intrusion. The surveillance missions are performed fully autonomously and are executed on-line and on board. The adopted architecture could also support coordination within multi-robot schemes through module *Communications*.

The setting of *IM* is dependent on the complexity, type, and object-density of the scenario, which in a dynamic environment can be different from one surveillance mission to another. To cope with this, all data (events and images) gathered in each tour are logged and off-line processed after the tour: the images are processed by *Object Detector* and, the events, by *IM*. The cases where *Object Detector* and *IM* disagree are submitted to a human operator for decision. This procedure enables estimating the *performance* of *IM* with the current parameters and, if necessary, a new *Auto-tuning* is triggered updating the training data set with the data obtained in the last tour. The execution of the training stage is off-line, on the *Ground Station*, and autonomous except for the cases in which the processing of the training data with the image-based *Object Detector* and event-based *IM* disagree.

IV. ASYNCHRONOUS EVENT-BASED INTRUSION MONITORING SYSTEM

Event cameras asynchronously generate events with µs resolution. Each event is described by $\mathbf{e} = (t, u, v, p)$, where (u, v) are the pixel coordinates, p is the event polarity (i.e., either 1 or 0), and t is the timestamp in which the event was triggered. Events are triggered by changes of intensity in the scene, which in general are caused by: (i) camera

motion producing changes of intensity from static objects in the scene, (ii) sensor inherent noise or due to manufacturing irregularities, and (iii) intensity changes from moving objects. Intruders in motion originate nearby corners in the event stream, e.g., caused by limbs in human and animal intruders, or by other robots. Hence, intruders create groups of events close to corners with globally consistent motion in the scenario. *Intrusion Monitoring* (*IM*) includes asynchronous event-by-event processing techniques for: corner detection, feature tracking, clustering, and also mechanisms to distinguish events corresponding to moving objects from those originated by the scene background. *IM* receives as input the adjusted event stream from ASAP [31], and provides as output the centroid of the detected intruders.

The operation of *IM* is as follows. The *Corner Detector* module implements the FA-Harris asynchronous event-based corner detector [33], selected due to its good performance in terms of false positive rate and computational cost when compared to others such as [19] and [21]. Both capabilities are crucial for real time applications in cluttered environments. Next, the corners with consistent motion are asynchronously tracked by the Feature Tracking module. Although some asynchronous corner trackers have been proposed [22], we preferred to develop a method adapted to the intrusion monitoring problem with improved candidate search and evaluation that trades between computational cost and accuracy, see Section IV-A. Even though corner tracks are useful to detect the intruder, they do not always provide enough information. The proposed *Clustering* algorithm creates clusters with the tracked features and nearby events with consistent motion, see Section IV-B. Besides, *IM* includes mechanisms to prevent processing events of static background caused by the camera motion. The proposed method models the spatio-temporal information in the event stream through the *Attention Priority Map* (*APM*), see Section IV-C, to define regions on the scene that trigger more events within a variable time window. Hence, regions with moving objects cause higher values in *APM*. Only the events which value in *APM* are higher than threshold ω are processed by the clustering algorithm. Only the clusters with sufficient number of events and corners are considered to be caused by an intruder. The rest are discarded. Finally, the bounding boxes and centroids of the resulting clusters are extracted and tracked. Algorithm 1 summarizes the operation of IM. Figure 3 shows experimental results from the execution of the above modules.

Algorithm 1 Asynchronous event-based intrusion monitoring

```
1:  procedure ASYNCHRONOUS EVENT-BASED INTRUSION MONITORING(e)
2:      isCorner ← CornerDetection(e)          ▷ Asynchronous corner detection.
3:      if isCorner then
4:          FeatureTracking(e)                 ▷ Asynchronous feature tracking.
5:      end if
6:      if not APMFiltered(e) then
7:          μ_c ←Clustering(e)                 ▷ Asynchornous clustering.
8:          CentroidTracking(μ_c)              ▷ Cluster centroid tracking.
9:      end if
10:     PackageProcessingAnalysis()            ▷ Feedback to ASAP.
11:     return μ_c                             ▷ Return tracked centroids.
12: end procedure
```

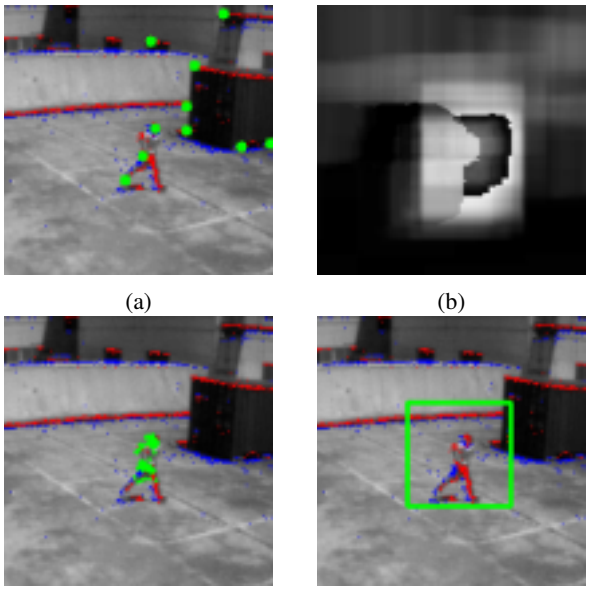

FIGURE 3: Results in different steps in the execution of *IM*: a) *Feature Tracking*, b) *APM*, c) *Clustering*, and d) *Centroid Tracking*. To facilitate visualization in a), c), and d), the obtained results are shown in green on the visual image provided by the APS, also showing in red and blue the events provided by the DVS during an equivalent time frame

Besides, *IM* includes several mechanisms to reduce its computational cost. One of them is the *ASAP* module [31], which adjusts the number of events provided to *IM* such that events are processed as soon as possible but avoiding processing overflow. The output of *IM* is continuously analyzed by *ASAP* (see line 10 in Algorithm 1) to enable online event-by-event processing. Besides, only events which priority in the *APM* is higher than a threshold ω are processed by the clustering algorithm. These mechanisms enable a large reduction of the computational cost without significantly reducing accuracy as is shown in Section VI.

The notation adopted in the *IM* modules is summarized in Table 1. *Feature Tracking*, *Clustering*, and *APM* employ similar symbols to represent buffers, lists, and timestamps, using different superindexes to distinguish between modules.

A. ASYNCHRONOUS EVENT-BASED FEATURE TRACKING

Algorithm 2 summarizes the asynchronous event-based feature tracker used in *IM*. The proposed method tracks features adopting an event-by-event asynchronous approach. Each event feature $\mathbf{f} = \{t, u, v\}$ is represented by its image coordinates $\mathbf{x} = (u, v)$ and the timestamp t of the event that generated the feature. In general events are sparsely distributed on the image, which hampers their tracking. To cope with this, our method makes use of a *Surface of Active*

Feature Tracking	
Symbol	**Description**
$\mathbf{S}$	SAeF.
$\mathbf{\Theta}^T$	List of features.
$\mathbf{f}$	Feature.
$\hat{\mathbf{f}}_i$	Feature track i.
$\mathbf{L}_\mathbf{f}^T$	Feature track candidate list of feature $\mathbf{f}$.
ν	Tracker proximity threshold.
$\mathbf{B}^T$	Event tracking buffer of size n^T.
τ^T	Oldest timestamp in $\mathbf{B}^T$.

APM	
$\mathbf{\Omega}$	Attention Priority Map.
$\mathbf{B}^A$	Event buffer for APM of size n^A.
τ^A	Oldest timestamp in $\mathbf{B}^A$.
l	Update window size.
ω	Filtering threshold.

Clustering	
Symbol	**Description**
c	Cluster.
$\mathbf{\Theta}^C$	List of clusters.
$\boldsymbol{\mu}_c$	Centroid of cluster c.
$\hat{\boldsymbol{\mu}}$	Cluster moving average.
$\mathbf{L}_\mathbf{e}^C$	Cluster candidate list for event $\mathbf{e}$.
r	Clustering proximity threshold.
$\mathbf{B}^C$	Event clustering buffer of size n^C.
τ^C	Oldest timestamp in $\mathbf{B}^C$.
$\mathbf{L}_c^E$	List of events associated to cluster c.
κ	Cluster proximity sampling number.

TABLE 1: Notation used in *Feature Tracking*, *Clustering*, and *APM*.

event Features (*SAeF*) that describes the spatio-temporal evolution of features over time. The *SAeF*, denoted as $\mathbf{S} \in \mathbb{R}^2$, is generated by spatially accumulating the last n^T detected features instead of using a fixed time window. The list of features represented on $\mathbf{S}$ is buffered in $\mathbf{B}^T$. Hence, the detection of a new feature triggers the update of $\mathbf{S}$ and $\mathbf{B}^T$ –adding the new feature and removing the oldest. As a result, $\mathbf{S}$ keeps a time-varying representation that adapts to the scene and camera motion. For instance, with fast camera motions the resulting $\mathbf{S}$ contains features detected in smaller time windows than with slow camera motions.

The input features are filtered to cancel noisy samples. A feature is considered as candidate for tracking if the number of neighboring features in $\mathbf{S}$ is greater than a threshold ν. Feature candidates are later used to either update or create new feature tracks. Hence, only features with frequent spatial occurrence are tracked. The set of feature tracks is stored in the list $\mathbf{\Theta}^T = [\hat{\mathbf{f}}_1, \dots \hat{\mathbf{f}}_n]$, where $\hat{\mathbf{f}}_i$ is the i-th feature track. The association between a new feature candidate $\mathbf{f}$ and existing feature tracks is as follows. Each feature candidate creates a list $\mathbf{L}_\mathbf{f}^T$ with the tracks it can be potentially associated to. $\hat{\mathbf{f}}_i$ is added to $\mathbf{L}_\mathbf{f}^T$ if its distance to $\mathbf{f}$, $D(\hat{\mathbf{f}}_i, \mathbf{f})$, is lower than a threshold d, which is typically small ($d = 5$ pixels in all the performed experiments) to avoid wrong associations with nearby objects in the scene. If$\mathbf{L}_\mathbf{f}^T$ has one only element $\mathbf{f}$, that track is updated with $\mathbf{f}$. If $\mathbf{L}_\mathbf{f}^T$ has more than one element, there exist several very close tracks –very likely redundant due to the low value of d adopted. In this case older tracks are prioritized since they represent more temporally consistent –and relevant for tracking– features over time. Hence, $\mathbf{f}$ is associated to the oldest track in$\mathbf{L}_\mathbf{f}^T$, denoted as $\hat{\mathbf{f}}^o$. The remaining tracks in $\mathbf{L}_\mathbf{f}^T$ are removed. Finally, if $\mathbf{L}_\mathbf{f}^T = \emptyset$, a new track is created and initialized with $\mathbf{f}$. Each time a feature track is updated, its timestamp is also updated. If the timestamp of a feature track is older than τ^T, the track is removed. τ^T is selected as the timestamp of the oldest feature in $\mathbf{B}^T$, enabling a time-varying forgetting reference that adapts to the scene and camera motion. It is preferred over a constant forgetting reference, which effectiveness is compromised by the camera motion, as reported in [34].

Algorithm 2 Asynchronous event-based feature tracking

procedure ASYNCHRONOUS EVENT-BASED TRACKING($\mathbf{f}$, ν)
  $\tau^T \leftarrow$ UpdateSurfaceOfActiveFeatures($\mathbf{f}$,$\mathbf{B}^T$,$\mathbf{S}$)
  if isCandidate($\mathbf{f}$,$\mathbf{S}$, ν) **then**
    $\mathbf{L}_\mathbf{f}^T \leftarrow$ SearchMatch($\mathbf{\Theta}^T$,D,$\mathbf{f}$) ▷ Find matches in $\mathbf{\Theta}^T$.
    if $\mathbf{L}_\mathbf{f}^T = \varnothing$ **then**
      $\mathbf{\Theta}^T \leftarrow$ Append($\mathbf{f}$) ▷ Add a new feature to $\mathbf{\Theta}^T$.
    else
      $\hat{\mathbf{f}}^\mathbf{o} \leftarrow$ GetOldestTrack($\mathbf{L}_\mathbf{f}^T$)
      $\mathbf{\Theta}^T \leftarrow$ Update($\mathbf{f}$,$\hat{\mathbf{f}}^\mathbf{o}$) ▷ Update $\hat{\mathbf{f}}^\mathbf{o}$ by $\mathbf{f}$ in $\mathbf{\Theta}^T$.
    end if
  end if
  $\mathbf{\Theta}^T \leftarrow$CleanTracker($\mathbf{L}_\mathbf{f}^T$, τ^T) ▷ Remove old features.
  return $\mathbf{\Theta}^T$ ▷ Return list of tracks.
end procedure

All the parameters of the tracker are set with fixed values except n^T, the size of buffer $\mathbf{B}^T$. As said above, d, the minimum feature-track association distance, is set to 5 as a trade-off between avoiding wrong associations from close tracks and dealing with the lack of continuity in the occurrence of the input features. Differently, the size of $\mathbf{B}^T$ should be chosen according to the camera, scenario camera motion and complexity. Static scenarios with few objects and slow camera motion can be represented with small buffers as few events are triggered under these conditions. However, cluttered scenes with dynamic objects require larger buffers. The setting of n^T is performed by the auto-tuning method described in Section V.

B. INTRUDER SEGMENTATION USING ASYNCHRONOUS EVENT-BASED CLUSTERING

An asynchronous event-based clustering is used to group the features and events caused by intruders. It employs the spatio-temporal proximity between events as grouping criteria, not requiring a-priori knowledge of the scene geometry or the number of objects in the scene. The computational capabilities on board an aerial robot is often limited. Unlike other clustering methods that aim for accuracy over efficiency, such as [23], our method includes a mechanism to balance efficiency and accuracy in order to avoid processing bottlenecks while keeping accuracy as high as possible.

The proposed clustering method is shown in Algorithm 3. Similarly to the tracker in Section IV-A, it keeps a buffer $\mathbf{B}^C$ with the n^C most recent events. $\mathbf{B}^C$ is updated with each new

event, adding the new event and removing the oldest. Also, τ^C is the timestamp of the oldest event in $\mathbf{B}^C$. It is used as a dynamic time horizon enhancing robustness to camera or scene motion. A cluster c is defined by: its centroid μ_c, a weighted moving average $\hat{\boldsymbol{\mu}}_c$ that is used for event-cluster association, and a list $\mathbf{L}_c^E$ of the events in $\mathbf{B}^C$ assigned to the cluster, i.e., the events assigned to c which timestamp is greater than τ^C. $\mathbf{L}_c^E$ is periodically updated by removing the events with timestamp lower that τ^C. Cluster c is deleted if all its assigned events are removed, i.e., if $\mathbf{L}_c^E = \emptyset$.

Event-cluster assignation is performed using proximity criteria. In general, events caused by an object are consistently triggered during shorts periods of time at specific parts of its contour. To capture this, $\hat{\boldsymbol{\mu}}_c$ is a weighted moving average that represents the centroid of the object contour. Every time a new event $\mathbf{e}$ is assigned to c, $\hat{\boldsymbol{\mu}}_c$ is updated with $\hat{\boldsymbol{\mu}}_c = \big(\alpha\mathbf{x} + (1-\alpha)\hat{\boldsymbol{\mu}}_c\big)/2$, where $\mathbf{x} = (u, v)$ are the coordinates of $\mathbf{e}$ and $\alpha \in [0, 1]$ is the weighting parameter typically selected as $\alpha > 0.5$ to give more weight to the new events. Hence, the proximity of a new event $\mathbf{e}$ to cluster c is evaluated using the distances from the event to the weighted average $\hat{\boldsymbol{\mu}}_c$ and to κ random samples drawn from $\mathbf{L}_c^E$. Cluster c is considered close to $\mathbf{e}$ if any of these distances is below a threshold r. The Manhattan distance was adopted as it is more efficient than the Euclidean distance and both obtained similar performance in the conducted experiments.

Each new event creates a list $\mathbf{L}_\mathbf{e}^C$ with the clusters it can be potentially assigned to. After all clusters have been evaluated, three cases are possible. If $\mathbf{L}_\mathbf{e}^C$ contains one only cluster, $\mathbf{e}$ is added to the cluster. If $\mathbf{L}_\mathbf{e}^C$ contains more than one cluster, the clusters are merged into one and $\mathbf{e}$ is added to the new cluster. If $\mathbf{L}_\mathbf{e}^C = \emptyset$, a new cluster is created with the event. Adding event $\mathbf{e}$ to a cluster c with n_c events involves: adding $\mathbf{e}$ to $\mathbf{L}_c^E$, and updating $\hat{\boldsymbol{\mu}}_c$ (as described above) and $\boldsymbol{\mu}_c$ as $\boldsymbol{\mu}_c = (n_c\boldsymbol{\mu}_c + \mathbf{x})/(n_c + 1)$. Similarly, when an old event is removed from $\mathbf{B}^C$, it should be removed from the cluster it is assigned to.

The number of samples κ in the cluster contour used to evaluate event-cluster proximity establishes a trade-off between computational cost and clustering accuracy. Hence, it should be chosen considering the limitations of the processing platform. In all experiments, κ was set to 100, which allowed on-line execution while providing good clustering performance. On the other hand, the size of buffer $\mathbf{B}^C$ and the radius r in event-cluster proximity evaluation are dependent on the scene and environment complexity. In dense scenes, small values of r are required to prevent wrong associations. In sparse scenes, high values of r will enhance the algorithm performance as the proximity to the cluster is found without evaluating all κ samples. Also, the size of $\mathbf{B}^C$ should be chosen according to the scene complexity. Simple scenes containing only one object on a uniform background can be represented with small buffers. Conversely, complex scenes require large buffers to accumulate the events corresponding to the scene details. The setting of r and size of $\mathbf{B}^C$ for the specific scenario is performed by the auto-tuning method described in Section V.

Algorithm 3 Asynchronous event-based clustering

procedure ASYNCHRONOUS EVENT-BASED CLUSTERING($\mathbf{e}$, r, κ)
  $\tau^C \leftarrow$ Update($\mathbf{e}$,$\mathbf{B}^C$) ▷ Update Event Buffer
  for each $c \in \boldsymbol{\Theta}^C$ **do**
    $\mathbf{L}_c^E \leftarrow$ Forget(c, τ^C) ▷ Remove old points from cluster.
    if Size($\mathbf{L}_c^E$) $= 0$ **then**
      $\boldsymbol{\Theta}^C \leftarrow$ Remove(c, $\boldsymbol{\Theta}^C$) ▷ Remove empty cluster.
      continue
    end if
    $\mathbf{L}_\mathbf{e}^C \leftarrow$ GetProximity($\mathbf{e}$,c,r,κ)
  end for
  if Size($\mathbf{L}_\mathbf{e}^C$) $= 0$ **then**
    $\boldsymbol{\Theta}^C \leftarrow$ CreateNew($\mathbf{e}$, $\boldsymbol{\Theta}^C$) ▷ Create new cluster.
  else if Size($\mathbf{L}_\mathbf{e}^C$) $= 1$ **then**
    $\boldsymbol{\Theta}^C \leftarrow$ AddTo($\boldsymbol{\Theta}^C$, $\mathbf{L}_\mathbf{e}^C$, $\mathbf{e}$) ▷ Add event to cluster.
  else
    $\boldsymbol{\Theta}^C \leftarrow$ MergeAndAdd($\mathbf{e}$, $\mathbf{L}_\mathbf{e}^C$) ▷ Merge clusters & add event.
  end if
  return $\boldsymbol{\mu}$ ▷ Return cluster centroid.
end procedure

C. ATTENTION FOCUS FOR ASYNCHRONOUS EVENT-BASED VISION

State-of-the-art event-based surveillance methods use static cameras [8], [9] assuming that the static background does not generate events. Such simplification cannot be assumed in case of event cameras on board UAS: even the residual UAS motions and vibrations during hovering can originate significant number of events. The proposed system requires methods capable of differentiating the events caused by static objects due to the robot motion from those originated by moving objects. To deal with that, the proposed method differentiates between regions with different event spatio-temporal density, assuming that, even with non-static cameras, moving objects cause significantly more events than the scene background. It is based on building a map that represents the regions on the scene that trigger more events within a variable time window, i.e., regions more likely to contain moving objects. Inspired by neuroscience, this map is denoted *Attention Priority Map* (*APM*). The *APM*, $\boldsymbol{\Omega} \in \mathbb{R}^2$, has the same resolution as the event camera, and is also updated asynchronously event by event. With each incoming event, $\boldsymbol{\Omega}$ is updated by increasing the values in a l-sized window centered at the event coordinates $\mathbf{x} = (u, v)$ as follows:

$$\boldsymbol{\Omega}_{ij} = \boldsymbol{\Omega}_{ij} + l - D(\mathbf{x}, \mathbf{y}) + 1, \tag{1}$$

where $D(\cdot)$ is the Manhattan distance, $\mathbf{y} = (i, j)$, $i \in [u - \frac{l-1}{2}, u + \frac{l-1}{2}]$, and $j \in [v - \frac{l-1}{2}, v + \frac{l-1}{2}]$.

$\boldsymbol{\Omega}$ is computed only with the last n^A events received. These events are temporarily stored in a buffer $\mathbf{B}^A$, where τ^A denotes the oldest timestamp in $\mathbf{B}^A$. $\boldsymbol{\Omega}$ is kept updated: events that become old are removed from $\mathbf{B}^A$ and from $\boldsymbol{\Omega}$ similarly as in Equation (1). $\boldsymbol{\Omega}$ is kept normalized within the range $[0, 1]$. An incoming event at coordinates (u, v) is considered to attract attention for intruder detection if $\boldsymbol{\Omega}(u, v)$ is greater or equal than a threshold $\omega \in [0, 1]$. Figure 4 shows an event

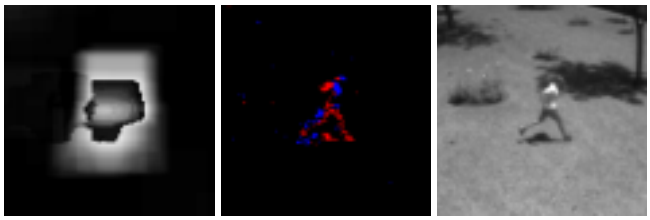

FIGURE 4: Example of the operation of *APM*: left) resulting $\mathbf{\Omega}$, center) events filtered by *APM*, and right) corresponding visual image.

image and the corresponding $\mathbf{\Omega}$ in an example with a person moving in the scene. Figure 4-left shows the resulting $\mathbf{\Omega}$ with the areas of highest values represented in white.

The performance of *APM* depends on three parameters: ω, which determines the sensitivity threshold in $\mathbf{\Omega}$ to pay attention to a moving object; l, the size of the window in the building of $\mathbf{\Omega}$; and n^A, the size of buffer $\mathbf{B}^A$. The type and complexity of the scene impact on the spatio-temporal density of the generated events and hence, on the selection of values for ω, l, and n^A. For instance, sparse and simple scenes can be represented using a low-size buffer $\mathbf{B}^A$ (i.e., low values of n^A), while complex scenes require larger buffers to support the scene complexity. Besides, using high-size buffers in sparse scenes can result in poor representation of the scene dynamics and unrealistically-large areas of attention. Similar dependencies with the scene type and complexity can be found for ω and l. Manually tuning these parameters for a given problem is not straightforward: they are adjusted automatically by the proposed auto-tuning system presented in Section V.

V. INTRUSION MONITORING AUTO-TUNING

A suitable tuning of parameters of Machine Learning (ML) and computer vision algorithms is crucial to ensure the desired performance, and in some cases, to the reduce computational cost. Empirical or manual parameter selection leads to long inefficient trial-and-error iterative processes that become more complex as the number of parameters increases.

This section proposes a method to tune the parameters of *IM*. Only the parameters that are dependent on the surveillance scenario and mission conditions, such as the scene complexity and sparseness/density, or the size of the objects in the scene, are tuned. Non-scene-dependent parameters, such as κ, ν, d, or α, were set with the same values in all the conducted experiments as described in previous sections. Automatically tuning non-scene-dependent parameters would unnecessarily increase the size of the search space, hampering and adding computational cost to the optimization process. The parameters that are tuned are: n^T, ω, l, n^A, r, and n^C. Their dependencies with the scenario were discussed in Section IV. Additionally, the threshold λ is considered for tuning. λ is defined as the minimum number of events per cluster to consider it as an intrusion detection. Event rate increases with faster camera motions and with lower object-camera distances. Hence, choosing an adequate value of λ depends on the surveillance scenario and mission. The parameters to be automatically tuned are grouped in parameter set $\mathbf{v} = [n^T, \omega, l, n^A, r, n^C, \lambda]$, see Table 2. The search space of each parameter is also shown in Table 2.

Parameter	Module	Description	Search space
n^T	Feature tracking	Size of buffer $\mathbf{B}^T$	$[50, 200]$
ω	*APM*	Sensitivity threshold to pay attention to a moving object	$[0, 1]$
l	*APM*	Size of the window used in the building of $\mathbf{\Omega}$	$[5, 40]$
n^A	*APM*	Size of buffer $\mathbf{B}^A$	$[50, 500]$
r	Clustering	Radius in event-cluster proximity evaluation	$[1, 50]$
n^C	Clustering	Size of buffer $\mathbf{B}^C$	$[20, 200]$
λ	Clustering	Minimum number of events per cluster to consider it an intrusion	$[0, 100]$

TABLE 2: Parameters of *IM* that are auto-tuned and their search space.

A. SIMULATED ANNEALING

Simulated Annealing (SA) [35] was adopted for parameter auto-tuning. SA is a well-known probability-based meta-heuristic algorithm that approximates the global optimum for the set of parameters in a large search space. Unlike gradient-based optimization methods, SA provides natural robustness against local minima and is suitable for cases when the function to be optimized is unknown but can be evaluated, see e.g., [29], as in the described problem. Also, besides finding a precise optimum if executed during enough time steps [36], SA is suitable in cases where finding an approximation to the global optimum in a limited time is preferable to finding a very precise optimum in a larger possibly-unfeasible time. This is the case since if necessary, *IM* parameters should be re-trained between two consecutive surveillance tours. Finally, it should be noted that the proposed scheme for parameter auto-tuning is flexible and, although out of the scope of the paper, other black-box optimization methods could be used.

Along the iterations, SA uses the current solution $\hat{\mathbf{v}}$ as reference to explore other solutions. In the early iterations, it allows the exploration of distant points in the search space by accepting high probability solutions that do not improve the current solution. At later iterations, it explores at shorter distances in the search space only accepting solutions that improve the current solution. At each iteration i, SA adds a perturbation $\Delta\mathbf{v}$ to the current solution $\hat{\mathbf{v}}$ and evaluates the cost function $J(\cdot)$ of the candidate solution $\mathbf{v}$, see Section V-B. The perturbation is sampled from a Gaussian distribution $N(0, \sigma\Upsilon_i)$, where σ is the perturbation standard deviation, and Υ_i is the temperature parameter that decreases over time controlling the annealing process. SA accepts candidate solutions that improve the cost function and also implements

a probabilistic mechanism to accept candidate solutions that do not improve it in order to prevent local minima. A candidate solution $\mathbf{v}$ is accepted according to a probability obtained from the Boltzman distribution as $p = e^{-\frac{\Delta J}{\Upsilon_i}}$, where $\Delta J = J(\hat{\mathbf{v}}) - J(\mathbf{v})$ is the cost difference between the candidate solution $\mathbf{v}$ and the current solution $\hat{\mathbf{v}}$. For great values of Υ_i, p tends to 1, which favors the acceptance of candidates with cost higher than $J(\hat{\mathbf{v}})$, enabling exploring distant points in the search space. Differently, for small values of Υ_i, p tends to 0, and the search is refined locally by accepting only the candidate solutions that improve $J(\hat{\mathbf{v}})$. The adopted annealing control followed a geometric model $\Upsilon_i = \Upsilon_0 \eta^i$, where $\eta \in [0.7, 0.96]$ defines the annealing relation. This geometric model was preferred over other annealing control schemes (e.g., linear) due its better experimental results in all the experiments performed.

B. PARAMETER AUTO-TUNING

The SA optimization process aims at maximizing the similarity of the result provided by the *IM* to a ground truth reference. To minimize the involvement of human operators, the ground truth is taken as the segmented objects classified as person provided by an automatic detector fed with visual images of the APS sensor of the DAVIS. YOLO V3 [32] was employed due its well-known fast-response object detection and prediction capabilities. Also, it can process the whole image instead of sliding windows or regions, which is interesting to consider the effect of the background. The scheme of the proposed auto-tuning method is summarized in Figure 2. The *Object Detector* module receives visual images and returns as output bounding boxes with the detected objects. The *IM* receives as input the event stream and returns the centroid of the detected intrusions. In the *Synchronizer* module both outputs are compared to evaluate the performance of *IM* leading to four possible outcomes: false positive, when *IM* reports an intrusion that is not reported by the object detector; true positive, when both object detector and *IM* report an intrusion and the distance between both centroids is lower than a threshold; true negative, when both *IM* and object detector report no intrusion; and false negative, when the object detection reports an intrusion and *IM* does not, or when the distance between the centroids of both reported intrusions is greater than the threshold.

The training set is composed of episodes, sequences of images and events gathered by the DAVIS during UAS flights in the intrusion monitoring scenario. Episodes were preferred over single images and their equivalent batch of events since they are more suitable to capture the scene dynamics and complexity. The training episode set should contain a sufficient number of episodes that cover the range of conditions where *IM* should operate including lighting conditions, object density and size, and presence of intruders, among others. At each SA iteration, the *Synchronizer* estimates the performance of *IM* configured with parameter set $\mathbf{v}$ by employing its intrusion detection accuracy:

$$A(\mathbf{v}) = \frac{TP(\mathbf{v}) + TN(\mathbf{v})}{TP(\mathbf{v}) + FP(\mathbf{v}) + TN(\mathbf{v}) + FN(\mathbf{v})}, \quad (2)$$

where $FP(\mathbf{v})$, $TP(\mathbf{v})$, $TN(\mathbf{v})$, and $FN(\mathbf{v})$ are respectively the number of false positives, true positives, true negatives, and false negatives obtained by *IM* configured with parameter set $\mathbf{v}$ in the training episode set.

Parameter Tuning implements the SA algorithm and evaluates the parameter set $\mathbf{v}$ using cost function $J(\mathbf{v}) = 1 - A(\mathbf{v})$. At each iteration, *IM* and *Object Detector* processes the episodes in the training set. *Syncronizer* computes $A(\mathbf{v})$, which is used by Parameter Tuning to obtain the updated parameter set $\hat{\mathbf{v}}$.

VI. EXPERIMENTAL VALIDATION AND ANALYSIS

The proposed scheme was validated and evaluated in sets of experiments conducted in realistic scenarios. The scheme operation and its performance in daylight conditions is summarized in Section VI-A. Section VI-B evaluates its performance adopting different auto-tuning approaches. Finally, its performance in dark conditions and robustness against lighting conditions is analyzed in Section VI-C.

The aerial platform used in the conducted experiments, see Figure 5, was a custom-made frame endowed with a PixHawk 1 autopilot running *PX4* position-based low-level controller, a U-Blox GPS receiver, a *DAVIS 346* event camera mounted at -45^o pitch rotation, and an INTEL® NUC6i7KYK2 embedded computer for on-line onboard computation and data logging. The navigation system was implemented on top of the UAL abstraction layer [37]. The proposed scheme was implemented in C++ using ROS Kinetic. The *IM* parameter auto-tuning system was performed off-line in an external computer with an AMD Ryzen 5 2600 processor and an NVDIA GeForce GTX 1070 Ti GPU.

FIGURE 5: Aerial platform used in the experiments.

The experiments consisted in evaluating the presence of intruders at different zones of the monitoring area, and were conducted around the laboratories of the School of Engineering of the University of Seville. A set of surveillance missions with different UAS trajectories and waypoints was defined. The zones (each of them defined by a 6-DoF waypoint in the

FIGURE 6: Example of surveillance mission used in the system experimentation conducted at the laboratories of the School of Engineering of the University of Seville. The frames represent the 6 DoF waypoints in the surveillance mission. The blue circular sectors represent the field of view of the event camera at each waypoint.

UAS trajectories) were selected with no special care, only determined by the intrusion monitoring application needs. Figure 6 shows one of these surveillance missions where the UAS transverses 420 m (go and back) approx. It includes 15 zones with high diversity in object density and type, object size and camera distance to objects, or intrinsic movement of some objects, such as trees, among others, and for this reason this mission is taken to illustrate the performance of the proposed scheme. The trajectory followed by the UAS in the execution of one mission is shown in red, and the event camera field of view in each waypoint, in blue.

First, prior flights were performed in different conditions (different day and night times, lighting conditions, and weather conditions, among others) to collect data for parameter auto-tuning. The robot started the trajectory and, when it reached one waypoint, it stayed in hovering and recorded 5 sequences of $10s$ with the collected events and images from the DAVIS sensor. To prevent bias in parameter training, 50% of the flights were recorded with people acting as intruders, and the rest, with no intruders.

After parameter set training, the scheme was evaluated in surveillance missions where the robot autonomously followed the trajectory. At each waypoint, the robot stayed in hovering while it executed on-line and on-board the *IM* event-based processing system configured with the trained parameter set. Some results from different steps in the event-based processing were shown, for brevity interleaved with the algorithm description, in Figures 1, 3, and 4. A video with some of the results is available as supplemental material. The scheme performance was assessed using the following well-known metrics [38]:

$$Accuracy = \frac{TP + TN}{P + N}, \tag{3}$$

$$Precision = \frac{TP}{TP + FP}, \tag{4}$$

$$TPR = \frac{TP}{TP + FN}, \tag{5}$$

$$FPR = \frac{FP}{FP + TN}, \tag{6}$$

where P is $TP + FN$, N is $TN + FP$, and TP, TN, FP, and FN stand respectively for the true positive, true negative, false positive, and false negative rates. The above metrics are expressed in percentages.

In the performed experiments *IM* required an average of $2.71\mu s$ to process every event. Besides, as described above, the different modules in *IM* include filtering effects. For example, *APM* discarded 59% of the events as originated by the background; these events were not processed by *Clustering*, which represents $\sim$70% of the processing time of *IM*. Moreover, *IM* processes the events sent by *ASAP*, which dynamically adjusts the number of events that are processed by *IM* to avoid processing overflow. In the experiments, the average event rate was $\sim$ $625,000$ events per second. The combined filtering effect of the different implemented modules and the fact that the corner detector used can be executed in real-time [33], allowed the processing hardware to execute the proposed scheme on-line and on-board.

A. PERFORMANCE EVALUATION

The zones considered were very diverse in object density, size of objects, or intrinsic movement of some objects (e.g., trees), see for instance Figure 7. The first analyzed approach was to capture the particularities of each zone in a specific parameter set trained only with data from that zone. This approach uses *zone-dependent* auto-tuned parameters where the robot changes its *IM* parameter set from one zone to another.

For all zones, the SA optimization was set with $\Upsilon_0 = 1.0$, $\eta = 0.955$, and $\sigma = 0.35$, which were chosen empirically to bring Υ_i close to zero in a fixed number of epochs. Iterating through too many epochs might require an unreasonable amount of time, whereas too low iteration numbers might cause inaccurate parameter tuning. We found that 150 epochs were a suitable trade-off between training time and expected accuracy for all the experiments. Each epoch corresponds to the processing of a sequence of $10s$.

The auto-tuning process started with an initial parameter set, which values were randomly selected within their bounds defined in Table 2. At each iteration, the cost function $J(\cdot)$ was evaluated for the candidate parameter set $\mathbf{v}$ and used to compute ΔJ. Then, a new parameter set $\mathbf{v}$ was generated in order to further search for a configuration that reduced the

FIGURE 7: Visual images of each of the 15 zones (ordered left to right and top to bottom) in the surveillance mission shown in Figure 6.

cost function. Figure 8 shows the values of $J(\hat{\mathbf{v}})$ and $J(\mathbf{v})$ during parameter training for zone 7. $J(\mathbf{v})$ describes the cost obtained for each configuration of $\mathbf{v}$ along the exploration of the search space. On the other hand, $J(\hat{\mathbf{v}})$ represents the cost obtained with the current set of parameters $\hat{\mathbf{v}}$, which at each iteration are used as prior to generate the next set of candidates $\mathbf{v}$. Figure 9 shows the evolution of $J(\hat{\mathbf{v}})$ during the auto-tuning process for each zone of the monitoring area. Due to the random initial parameter set and the different characteristics in each zone, the cost obtained at the first epoch varied among the different zones. As pointed out in Section V-A, the cost $J(\hat{\mathbf{v}})$ could increase during the first iterations of the training (when Υ_i is close to 1) to enlarge the exploration of the search space and prevent local minima. After 20 epochs the cost function was below 0.2 for all zones considered, showing fast convergence. After 150 epochs, when the training was completed, the cost function reached values below 0.1 for all zones.

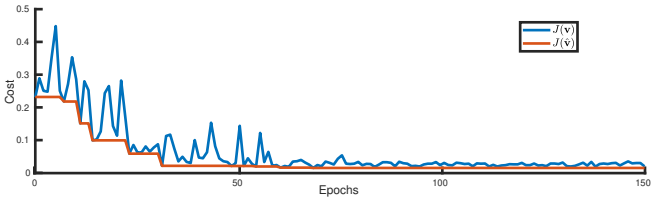

FIGURE 8: Values of cost functions $J(\hat{\mathbf{v}})$ and $J(\mathbf{v})$ during the parameter training for zone 7. $J(\mathbf{v})$ describes the performance of the different solutions obtained along the exploration of the search space

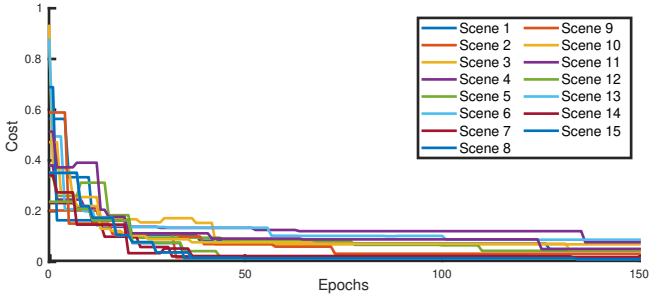

FIGURE 9: Evolution of the values $J(\hat{\mathbf{v}})$ during the *zone-dependent* training for each zone.

After training, more than 30 evaluation missions were performed in different conditions including different day hours, lighting, and meteorological conditions. Table 3 shows the average performance metrics in each zone obtained in the evaluation missions. The trained scheme provided high $Accuracy$ in all zones, with an average of 96%. Further, the values of $Accuracy$ and FPR showed a low number of "false alarms", evidencing high false positive rejection capability. Also, TPR showed reasonable capabilities regarding missing detections.

B. ANALYSIS OF ZONE SENSITIVITY

This section evaluates the proposed scheme using the same parameter set for all zones in the monitoring mission. The naïve approach of using one parameter set $\mathbf{v}'$ computed as the mean, median, or statistical fashion of all the independently-tuned parameter sets (one for each zone) would work only for those zones which parameters were close to $\mathbf{v}'$. Hence, we opted for training a single *zone-independent* parameter

Zone	*Accuracy* (%)	*Precision* (%)	*TPR* (%)	*FPR* (%)
1	95	97	87	1
2	98	97	96	1
3	96	90	90	5
4	92	94	85	3
5	96	96	91	2
6	91	92	83	4
7	98	98	97	1
8	98	98	98	1
9	95	92	94	3
10	97	94	97	3
11	95	96	92	2
12	95	96	90	1
13	99	98	98	1
14	98	98	95	1
15	97	98	95	1

TABLE 3: Performance evaluation in each zone using *zone-dependent* auto-tuned parameters.

set using a training set that includes sequences taken from all zones in the mission. At each epoch several randomly chosen sequences from different zones were used for the evaluation of the cost function. The SA optimization was set with $\Upsilon_0 = 1.0$, $\eta = 0.955$ and $\sigma = 0.35$, the same values as in Section VI-A.

The performance results obtained are reported in Table 4. Two types of results are shown: left) evaluation using the *zone-independent* approach, i.e., using one parameter set trained for all zones; and right) evaluation using the *parameter set averaging* approach, i.e., using for all zones the average of the parameter sets specifically trained for each zone. In general, the *Accuracy* values obtained by *zone-independent* training were 5% greater for all zones than those obtained by *parameter set averaging*. The performance of *parameter set averaging* was satisfactory for some zones, but, poor in others, such as zones 3 and 4, which had high complexity and differences w.r.t. the other zones. On the contrary, the *zone-independent* approach reported *Accuracy* values higher than 80% for all zones and an average *Accuracy* of 91%. Although the performance of the *zone-independent* approach was lower than that of the *zone-dependent* approach, it was still satisfactory and involved a significantly simpler and more efficient training process, favoring scalability to larger surveillance areas. It is worth considering that, finding a single parameter set that results in high *Accuracy* for every zone might not be possible in all problems, due to the diversity of the zones in the surveillance mission. Hence, *zone-independent* training is proposed as a trade-off solution when all the zones of a surveillance mission are relatively homogeneous, whereas *zone-dependent* training should be used otherwise.

C. PERFORMANCE UNDER LOW ILLUMINATION CONDITIONS

The results in Sections VI-A and VI-B show the performance of the proposed scheme in experiments performed during the day with different lighting conditions. This section analyzes its robustness in experiments conducted during the night with low and pitch dark lighting conditions.

First, the scheme using the *zone-independent* parameter set trained with daylight data (described in Section VI-B) was evaluated in flights performed under dark lighting conditions. Table 5-top shows the average performance obtained. The resulting *Accuracy* values in the different zones ranged between 73% and 84%. They were significantly lower than those obtained in the daylight experiments. Also, *TPR* performed worse than in daylight experiments due to the high number of false negatives caused by the level noise increment in event streams taken with dark lighting conditions. Besides, although *Precision* and *FPR* performed well, these results do not provide a complete measurement of performance as can be seen in Eqs. (4) and (6), both of them are independent of the number of false negatives. The obtained results suggest that a different set of parameters are needed to satisfactorily operate with dark lighting conditions.

To evaluate that hypothesis, a parameter set using only data collected during the night was trained by adopting the *zone-independent* approach. YOLO V3 did not provide sufficient reliability for detecting the intruders in dark lighting conditions, thus data were manually annotated. Table 5-center shows the average performance metrics obtained in evaluation surveillance missions also performed during the night. The values of *Precision*, *TPR*, and *FPR* were close to the values obtained during the day (Section VI-A), but the average *Accuracy* value was slightly lower due to the greater number of false alarms caused by the lower signal-to-noise ratio in dark scenarios. The noise increment affected the operation of the *APM* by reducing the priority of moving intruders. Although these results are acceptable, manual annotation is a very time-demanding task.

To cope with these issues, we analyzed the option of establishing a parameter set for dark lighting conditions by adapting the values of the parameter set trained for daylight conditions. To understand the intuition of the problem, datasets of different zones taken in daylight and dark lighting conditions were extensively analyzed. First, in all zones the number of events generated in dark lighting conditions were at least twice greater than those generated with daylight conditions. Many of them resulted from the event over-generation caused by edges, whereas others resulted from the noise level increment due to the low lighting conditions. This phenomenon was also reported in [14]. Furthermore, the parameter sets trained for daylight and dark lighting conditions in different zones were also analyzed. The main changes occurred in only three parameters: ω, n^C, and n^A. The values of ω, the thresholds in the *APM*, $\sim$30% from daylight to dark lighting conditions. Due to the higher noise level, the *APM* enhanced its filtering effect and imposed

	Zone-independent				*Parameter set averaging*			
Scene	*Accuracy (%)*	*Precision (%)*	*TPR (%)*	*FPR (%)*	*Accuracy (%)*	*Precision (%)*	*TPR (%)*	*FPR (%)*
1	91	99	72	1	91	97	74	1
2	95	96	84	1	92	76	95	9
3	80	84	60	7	67	54	62	30
4	86	96	66	2	73	60	83	32
5	88	98	74	1	86	89	73	6
6	87	96	68	2	81	71	81	19
7	96	96	94	2	90	79	98	14
8	97	99	93	1	95	90	97	7
9	93	92	88	4	94	88	97	7
10	92	96	84	3	86	80	89	15
11	88	96	75	2	86	88	74	6
12	93	98	85	1	82	97	60	1
13	99	98	98	1	98	97	97	1
14	93	98	81	1	89	98	70	1
15	92	96	85	1	91	96	81	1

TABLE 4: Evaluation in each zone of *IM* trained with the *zone-independent* (left) and with the *parameter set averaging* (right) approaches.

Parameter set	*Accuracy (%)*	*Precision (%)*	*TPR (%)*	*FPR (%)*
Parameters trained with daylight data	82	98	62	2
Parameters trained with dark lighting data	94	94	92	5
Parameters trained with daylight data adapted to night conditions	92	93	89	5

TABLE 5: Performance of the proposed scheme in night experiments using parameters trained with: top) only daylight data, center) only dark lighting data, and bottom) daylight parameters adapted to dark conditions.

more restrictions to consider an event as originated by an intruder. Also, the values of n^A, the sizes of the*APM* buffers in dark lighting conditions, were four times greater than in daylight conditions. The increment in the events triggered in dark lighting conditions required greater buffers to represent the scene. Finally, n^C, the sizes of clustering buffers in dark lighting conditions, were 50% smaller than in daylight conditions. The higher noise level in dark lighting can cause clustering large regions in the image. Low values of n^C limit the size of these clusters.

Table 5-bottom shows the performance obtained using the parameter set trained with daylight data but adapting the values of ω, n^C, and n^A to dark lighting conditions. Although its performance is slightly worse than that obtained by training with dark lighting data, parameter adaptation provides a trade-off solution that keeps high performance in both daylight and dark lighting without requiring training for dark lighting conditions.

VII. CONCLUSIONS AND FUTURE WORK

This paper proposes an event-based processing scheme for intrusion monitoring with UAS. Event cameras are very interesting sensors for outdoor aerial robot applications. They provide high dynamic range, being highly robust to lighting conditions, and enabling day/night operation with the same event camera. Also, they provide very high temporal resolution, being insensitive to motion blur. Finally, they are small and have low power consumption.

The proposed scheme includes two main perception blocks. First, an asynchronous event-based processing system detects intrusions on-line using only the events. It includes several event-based algorithms: namely, corner detection, tracking, clustering, and event filtering through the attention priority map. All of them process the events one-by-one, resulting in a fully asynchronous processing system that exploits the advantages of the sequential nature of event cameras. The algorithms were endowed with mechanisms to reduce their computational cost in order to enable on-line onboard execution. The second block is an off-line semi-supervised training mechanism that adjusts the parameters of the event-based processing algorithms to a particular scenario and problem. It is based on an optimization process that maximizes the similarity between the output of the event-based processing and ground truth results obtained applying an object detector to the visual images provided by the APS sensor of the DAVIS camera. The proposed perception scheme was implemented in ROS, integrated in a fully autonomous aerial robot architecture, and extensively validated in challenging scenarios with a wide variety of lighting conditions, including day and night experiments in pitch dark conditions.

A limitation of the proposed scheme is that it does not identify the type of intruder (e.g., person, car, or a bike).

Intruder identification using event-based deep learning techniques is object of current research. In this work, the different signal-to-noise ratios of events triggered at day and night were addressed by scaling some parameters of the event-based processing system based on empirical evidence. Current work focuses on modeling the event generation under different lighting conditions from an analytical point-of-view. Furthermore, the proposed method assumes intruders moving at higher velocities than the UAS. Additional experimental evaluation where both UAS and intruder move at similar speeds is subject of further analysis. Finally, this work was developed in the context of the GRIFFIN ERC Advanced Grant, which objective is to develop autonomous flapping-wing robotic systems capable of navigating, perching, and manipulating objects. The advantages of event cameras overcome some of the limitations of flapping-wing robot perception caused by agile maneuvers and abrupt movements [39]. Future work will also extend the proposed scheme to be used on board a flapping-wing robot to detect and track moving targets during perching and flight.

ACKNOWLEDGMENT

The authors would like to thank Rafael Salmoral for his support in the hardware integration and helping with the hexarotor experiments.

REFERENCES

[1] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240× 180 130 db 3 μs latency global shutter spatiotemporal vision sensor," IEEE Journal of Solid-State Circuits, vol. 49, no. 10, pp. 2333–2341, 2014.

[2] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143 db dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS," IEEE Journal of Solid-State Circuits, vol. 46, no. 1, pp. 259–275, 2010.

[3] S. Chen and M. Guo, "Live demonstration: CeleX-V: a 1M pixel multi-mode event-based sensor," in IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2019, pp. 1682–1683.

[4] R. Berner, "Event-based vision sensor," May 14 2020, uS Patent App. 16/682,505.

[5] G. Gallego, T. Delbruck, G. M. 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, pp. 1–1, 2020.

[6] 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 International Conference on Robotics and Automation (ICRA), 2020, pp. 1651–1657.

[7] J. P. Rodríguez-Gomez, A. G. Eguíluz, J. R. Martínez-de Dios, and A. Ollero, "Asynchronous event-based clustering and tracking for intrusion monitoring in uas," in 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 8518–8524.

[8] Z. Jiang, P. Xia, K. Huang, W. Stechele, G. Chen, Z. Bing, and A. Knoll, "Mixed frame-/event-driven fast pedestrian detection," in 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 8332–8338.

[9] S. Barua, Y. Miyatani, and A. Veeraraghavan, "Direct face detection and video reconstruction from event cameras," in 2016 IEEE Winter Conference on Applications of Computer Vision (WACV), 2016, pp. 1–9.

[10] G. Gallego and D. Scaramuzza, "Accurate angular velocity estimation with an event camera," IEEE Robotics and Automation Letters, vol. 2, no. 2, pp. 632–639, 2017.

[11] T. Stoffregen, G. Gallego, T. Drummond, L. Kleeman, and D. Scaramuzza, "Event-based motion segmentation by motion compensation," in Proc. IEEE International Conference on Computer Vision, 2019, pp. 7244–7253.

[12] A. Mitrokhin, C. Ye, C. Fermüller, Y. Aloimonos, and T. Delbruck, "Ev-imo: Motion segmentation dataset and learning pipeline for event cameras," in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 6105–6112.

[13] J. Xu, M. Jiang, L. Yu, W. Yang, and W. Wang, "Robust motion compensation for event cameras with smooth constraint," IEEE Transactions on Computational Imaging, vol. 6, pp. 604–614, 2020.

[14] A. Mitrokhin, C. Fermüller, C. Parameshwara, and Y. Aloimonos, "Event-based moving object detection and tracking," in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 1–9.

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

[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] R. S. Dimitrova, M. Gehrig, D. Brescianini, and D. Scaramuzza, "Towards low-latency high-bandwidth control of quadrotors using event cameras," in 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 4294–4300.

[18] V. Vasco, A. Glover, and C. Bartolozzi, "Fast event-based harris corner detection exploiting the advantages of event-driven cameras," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 4144–4149.

[19] E. Mueggler, C. Bartolozzi, and D. Scaramuzza, "Fast event-based corner detection," in Proceedings of the British Machine Vision Conference (BMVC). BMVA Press, September 2017, pp. 33.1–33.11.

[20] R. Benosman, C. Clercq, X. Lagorce, S.-H. Ieng, and C. Bartolozzi, "Event-based visual flow," IEEE Transactions on Neural Networks and Learning Systems, vol. 25, no. 2, pp. 407–417, 2013.

[21] 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.

[22] ——, "ACE: An efficient asynchronous corner tracker for event cameras," in International Conference on 3D Vision (3DV), 2018, pp. 653–661.

[23] F. Barranco, C. Fermuller, and E. Ros, "Real-time clustering and multi-target tracking using event-based sensors," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 5764–5769.

[24] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-based visual inertial odometry," in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 5816–5824.

[25] 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 Intl. Conf. on Intelligent Robots and Systems (IROS), 2020, pp. 5978–5985.

[26] B. Karasulu and S. Korukoglu, "A simulated annealing-based optimal threshold determining method in edge-based segmentation of grayscale images," Applied Soft Computing, vol. 11, no. 2, pp. 2246–2259, 2011.

[27] S. Yahyanejad and J. Ström, "Removing motion blur from barcode images," in IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops, 2010, pp. 41–46.

[28] A. Barbu, Y. She, L. Ding, and G. Gramajo, "Feature selection with annealing for computer vision and big data learning," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 2, pp. 272–286, 2016.

[29] R. S. Tavares, T. Martins, and M. d. S. G. Tsuzuki, "Simulated annealing with adaptive neighborhood: A case study in off-line robot path planning," Expert Systems with Applications, vol. 38, no. 4, pp. 2951–2965, 2011.

[30] J. R. Martínez de-Dios, F. J. Pérez-Grau, A. Torres-González, J. J. Acevedo, J. L. Paneque, A. Viguria, D. Fuego, J. R. Astorga, and A. Ollero, "GRVC-CATEC: Aerial robot co-worker in plant servicing (ARCOW)," in Bringing Innovative Robotic Technologies from Research Labs to Industrial End-users. Springer, 2020, pp. 211–242.

[31] 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.

[32] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," arXiv preprint arXiv:1804.02767, 2018.

[33] R. Li, D. Shi, Y. Zhang, K. Li, and R. Li, "Fa-harris: A fast and asynchronous corner detector for event cameras," in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 6223–6229.

[34] J. Wu, K. Zhang, Y. Zhang, X. Xie, and G. Shi, "High-speed object tracking with dynamic vision sensor," in China High Resolution Earth Observation Conference. Springer, 2018, pp. 164–174.
[35] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," science, vol. 220, no. 4598, pp. 671–680, 1983.
[36] X.-S. Yang, Nature-inspired optimization algorithms. Elsevier, 2014.
[37] F. Real, A. Torres-González, P. Ramón-Soria, J. Capitán, and A. Ollero, "Unmanned aerial vehicle abstraction layer: An abstraction layer to operate unmanned aerial vehicles," International Journal of Advanced Robotic Systems, vol. 17, no. 4, p. 1729881420925011, 2020.
[38] T. Fawcett, "An introduction to ROC analysis," Pattern recognition letters, vol. 27, no. 8, pp. 861–874, 2006.
[39] A. G. Eguíluz, J. Rodríguez-Gomez, J. Paneque, P. Grau, J. R. Martínez-de Dios, and A. Ollero, "Towards flapping wing robot visual perception: Opportunities and challenges," in 2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS). IEEE, 2019, pp. 335–343.

JUAN PABLO RODRÍGUEZ-GÓMEZ received his B.Sc. in Electronic Engineering from Pontifical Javeriana University - Bogotá (Colombia), and a MSE degree in Artificial Intelligence and Robotics from Sapienza University of Rome (Italy). Currently he is pursuing a Ph.D in Robotics with the GRVC Robotics Laboratory of the University of Seville (Spain). His research interests include event-based computer vision, robot perception, robot navigation, and machine learning.

AUGUSTO GÓMEZ EGUÍLUZ is a postdoctoral researcher with the GRVC Laboratory at the University of Seville (Spain). He received his BSc in Computer Science at the Pontifical University of Salamanca (Spain), and a MSc degree in the University of Salamanca (Spain). In 2018, he obtained his PhD with the Cognitive Robotics Team at the Ulster University (UK). His thesis focused on exploring tactile sensing as a continuous process to enhance robot collaboration capabilities for object manipulation. Nowadays, his research with the GRVC Laboratory explores the use of event-based vision in Unmanned Aerial Systems. His main research interests include robotic perception, machine learning, robotic manipulation, tactile sensing, and event-based vision.

J. RAMIRO MARTÍNEZ-DE DIOS is Full Professor at the University of Seville (Spain). His R&D activities are focused mainly on aerial robot perception, multi-robot cooperation, robot localization and mapping, and sensor fusion. He has authored or co-authored >130 publications on these topics, including 4 full books. He has also coordinated >14 R&D projects and has participated in other >60 R&D projects, including >18 projects funded by the European Commission in FPIV, FPV, FPVI, FP7, and H2020. He leaded and participated in >15 technology transfer contracts to companies such as AIRBUS, BR&TE, or IBERDROLA, among others. He has served as member of the Editorial Board of >7 journals, and TPC member in >45 conferences and workshops. He has received or co-received 5 international awards, including the "Best Drone-based Solution" award in the 1st EU Drone Awards.

ANIBAL OLLERO is Full Professor, the Head of GRVC, University of Seville, and a Scientific Advisor of the Center for Advanced Aerospace Technologies in Seville, Spain. He has been a Full Professor with the Universities of Santiago and Malaga, Spain, a Researcher with the Robotics Institute of Carnegie Mellon University, Pittsburgh, USA, and LAAS-CNRS, Toulouse, France. He authored more than 750 publications, including nine books and 200 journal articles and led about 160 projects, transferring results to many companies. He has participated in 25 European Projects being coordinator of seven, including the H2020 AEROARMS and H2020 AERIAL-CORE, and the ERC Advanced Grant GRIFFIN project. He has been a member of the Board of Directors of euRobotics until March 2019. He was a recipient of 14 awards, has supervised 43 PhD Thesis and is IEEE Fellow for contributions to the development and deployment of aerial robots. He is the Co-Chair of the IEEE Technical Committee on Aerial Robotics and Unmanned Aerial Vehicles, the Coordinator of the Aerial Robotics Topic Group de euRobotics. He has been also Founder and the President of the Spanish Society for the Research and Development in Robotics (SEIDROB), until November 2017.