Preprint version of:
Rodríguez-Gómez, J. P., Gómez Eguíluz, A., Martínez-de Dios, J. R., Ollero, A. (2020, June).
Asynchronous event-based clustering and tracking for intrusion monitoring in UAS.
In IEEE International Conference on Robotics and Automation (ICRA 2020)

Asynchronous event-based clustering and tracking for intrusion monitoring in UAS

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

***Abstract*— Automatic surveillance and monitoring using Unmanned Aerial Systems (UAS) require the development of perception systems that robustly work under different illumination conditions. Event cameras are neuromorphic sensors that capture the illumination changes in the scene with very low latency and high dynamic range. Although recent advances in event-based vision have explored the use of event cameras onboard UAS, most techniques group events in frames and, therefore, do not fully exploit the sequential and asynchronous nature of the event stream. This paper proposes a fully asynchronous scheme for intruder monitoring using UAS. It employs efficient event clustering and feature tracking modules and includes a sampling mechanism to cope with the computational cost of event-by-event processing adapting to on-board hardware computational constraints. The proposed scheme was tested on a real multirotor in challenging scenarios showing significant accuracy and robustness to lighting conditions.**

***Index Terms*— event camera, asynchronous, intrusion monitoring, surveillance, UAS, clustering, feature tracking.**

I. Introduction

Unmanned Aerial Systems (UAS) have been proposed as ideal tools for surveillance and monitoring of large areas in applications such as border surveillance [1] or intruder detection [2], among others. Automatic surveillance and intruder detection in large, complex and unstructured scenarios face relevant problems. Lighting conditions are a severe problem in automatic visual-based systems. Besides, different cameras are usually required for operating day and night, involving higher UAS payload, energy consumption or on-board computational needs and, hence, reducing the flight time. Motion blur is also a significant constraint in many UAS vision-based systems, e.g. in highly dynamic scenarios [3], new-generation aerial platforms, such as ornithopters [4] [5] [6], or simply in applications where poorly-maintained UAS originate strong mechanical vibrations.

In this paper, we apply event cameras for intruder monitoring. Event cameras provide high dynamic range and temporal resolution. They are insensitive to motion blur, lightweight and have low power consumption. A good number of successful techniques have been proposed in the last years evidencing the capabilities of event cameras [7]. The current trend in robotics is to group the received events in frames, i.e. *event images*. Processing of *event images* enables designing complex and elaborated techniques for computer vision in robotics. However, this approach does not always fully exploit the sequential and asynchronous capabilities of the data stream provided by event cameras. In fact, *event images* can create motion blur in some cases and some techniques, e.g. [8], include specific mechanisms to reduce it. Asynchronous event-by-event processing usually has higher computational needs and is mainly adopted for low-level processing techniques, e.g. feature detection [9] or tracking [10]. Some schemes combine asynchronous methods for low-level processing with others based on *event images* for high-level processing [11], but they do not always fully exploit either the advantages of event cameras.

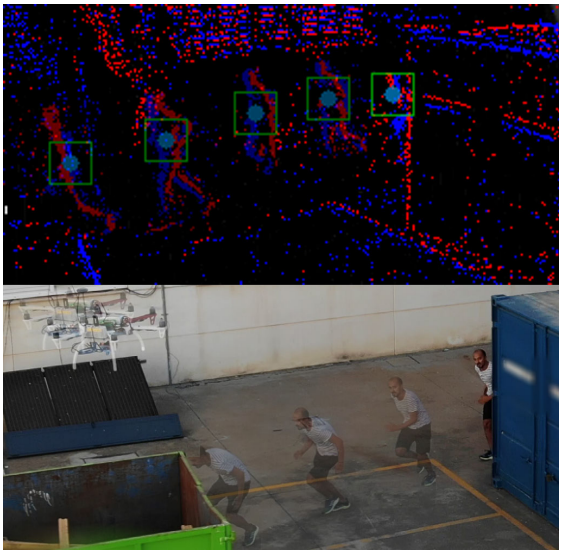


Fig. 1: Intrusion monitoring example of the proposed scheme in UAS-based experiments.

This paper presents an asynchronous event-based scheme for intrusion monitoring with UAS. It integrates efficient feature tracking and event clustering methods, among others. All of them perform event-by-event processing, resulting in a fully asynchronous processing. Special attention has been devoted to its design in order to enable on-line on-board execution. The proposed scheme has been implemented in ROS and validated in UAS-based experiments performed in complex and unstructured scenarios during the day and night with one and several intruders, see Figure 1.

The contribution of this paper is three-fold. First, a fully asynchronous scheme for intruder monitoring using UAS is presented. It can be tuned to run in real time adapting to the hardware computational constraints. Sec-

This work was supported by the European Research Council as part of GRIFFIN ERC Advanced Grant 2017, Action 788247 and ARM-EXTEND (DPI2017-8979-R) project funded by the Spanish National R&D Plan.
The authors are with the GRVC Robotics laboratory, University of Seville, Seville 41092, Spain email: {jprodriguez, ageguiluz, jdedios, aollero}@us.es

Preprint version of:
Rodríguez-Gómez, J. P., Gómez Eguíluz, A., Martínez-de Dios, J. R., Ollero, A. (2020, June).
Asynchronous event-based clustering and tracking for intrusion monitoring in UAS.
In IEEE International Conference on Robotics and Automation (ICRA 2020)

ond, two asynchronous methods for feature tracking and event clustering are presented. Third, code and datasets will be released in order to contribute to the development of event-based vision community (`https://grvc.us.es/davis-dataset-for-intrusion-monitoring/`).

The rest of the paper is organized as follows. Section II briefly summarizes the main works in the topics addressed in the paper. The proposed asynchronous event-based scheme for UAS surveillance is presented in Section III, along with its main components. Section IV presents the experimental validation and robustness analysis of the proposed scheme. Section V concludes the paper and highlights future research.

II. State of the art

The advent of event cameras has recently attracted significant research interest in the robotics and computer vision communities [7]. Some works have explored the use of event-based vision for surveillance tasks, mainly for pedestrian detection [12] [13]. The work in [12] proposed a method for face detection and high speed video reconstruction. A Bayesian inference approach was proposed in [13] to fuse the output of two YOLO classifiers fed with frames and *event images*. Both works assumed static cameras which severely constrains their use in robotics applications.

The use of event-based vision systems on real robots requires methods capable of coping with the camera motion. The work in [14] proposed a contrast maximization process to estimate the rotational motion of the camera using *event images* formed through polarity addition. The optical flow of the events was used in [15] to segment the scene by clustering motion-compensated images into objects according to their velocity. Motion compensation was also used in [16] to provide a cluster association for each event while estimating the motion parameters of the objects through an optimization process. Recently, the work in [17] developed the first approach for event-based independent motion detection using Neural Networks to estimate the camera egomotion and segment both depth and pixel motion. However, the performance of these methods onboard real robots was not evaluated.

A method to detect and track a moving circle (i.e. a ball) using a Hough transform approach and optical flow information from windows of a fixed number of events was implemented in an iCub robot [18]. Their method was later extended in [19] to enhance the tracking robustness using a particle filter. Another work using an iCub robot was presented in [11]. It detected and grouped corner features in order to predict the expected camera motion distribution as a function of the robot joint velocities using a ν-SVM. Their method performed asynchronously for corner detection and tracking while the independent motion detection was performed in a synchronous manner, i.e. the robot joint velocities were updated every 10 ms. A model of the affine transformation between two consecutive *event images* was used in [20] to compensate for the global motion of a Micro Aerial Vehicle (MAV) and, the resulting events were assumed to represent the moving objects. In [21], an autonomous MAV landing approach based on the optical flow of event frames obtained from a downwards-pointing DVS sensor was presented. Recently, the work in [8] proposed a high-speed dodging system for UAS. A Deep Learning solution used *event images* to detect independent moving objects, estimate their 3D motion, and avoid collisions.

Although the output of event cameras are asynchronous event streams, all the above techniques group the temporally-close received events in frames called *event images*. Hence, they do not fully exploit the advantages of event cameras and, in fact, some of them (e.g. [8]) include motion blur cancellation mechanisms. Various asynchronous event-by-event processing methods have been proposed for feature detection [9] [22] [23] [24], feature tracking [10] [25], clustering [26], pose tracking [27], and visual inertial odometry [28]. Particularly relevant for this paper is the asynchronous localization approach developed in [27] onboard a multirotor. The authors were capable of tracking the 6-DoF pose of the drone during high speed maneuvers by looking at a previously known planar shape on a wall. Although these works provide reliable solutions, their application on real robots navigating in realistic, complex and unstructured scenarios is still an under-researched area. This paper presents an asynchronous event-by-event processing scheme designed for and validated onboard a UAS in a realistic unstructured scenario.

III. The Proposed Method

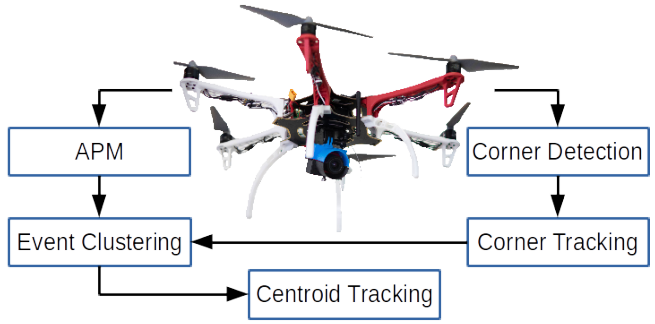
We are interested in monitoring intruders in complex and unstructured scenarios using UAS equipped with event cameras. The scenarios are assumed static but intruders move in the scenario. That is the case in many applications, e.g. night surveillance in factories or perimeter monitoring. The event cameras on moving UAS generate events from static objects in the scenario but the events generated by the intruders can be distinguished due to their different spatio-temporal properties. We assume that intruders originate nearby corners in the event stream, e.g. caused by limbs in human and animal intruders, or by ground-aerial robots. Hence, intruders create groups of events close to corners with globally consistent motion in the scenario.

The diagram of the proposed scheme is shown in Figure 2-a. All the modules in the scheme perform event-by-event processing, resulting in a fully asynchronous scheme, which can fully exploit the sequential nature of events. The events from the Dynamic Vision Sensor (DVS) of a DAVIS 346 camera onboard the UAS are received asynchronously. Event cameras use Address Event Representation (AER) to return information based on pixel intensity variation with a resolution of μ seconds. Each event is defined by $\mathbf{e} = (t, u, v, p)$, where t is the time in which the event was triggered, (u, v) are the pixel coordinates and p is the polarity of the brightness change, i.e. either 1 or 0.

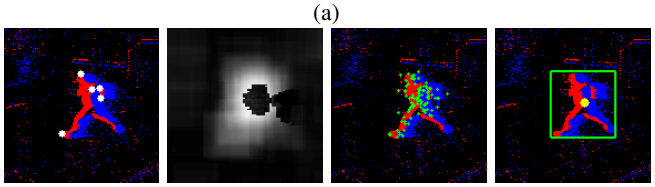
The first module in the scheme performs corner detection. Among the corner detection methods available online we adopted *eFast [24], a modified version of the method described in [29] due to its compromise between accuracy,

Preprint version of:
Rodríguez-Gómez, J. P., Gómez Eguíluz, A., Martínez-de Dios, J. R., Ollero, A. (2020, June).
Asynchronous event-based clustering and tracking for intrusion monitoring in UAS.
In IEEE International Conference on Robotics and Automation (ICRA 2020)

false positive rate, and computational efficiency [9]. Then, the corners detected are separated by polarity and tracked to remove inconsistent and noisy corners. An example of the corner tracking output is shown in Figure 2-b. Although some asynchronous corner tracker methods have been proposed [10] [30], we preferred to develop a new method –described in Section III-A, adapted to our problem and more efficient.



(a)



(b) (c) (d) (e)

Fig. 2: (a) Scheme of the proposed asynchronous scheme. Results from each developed module: b) corner tracking, c) APM, d) event clustering and, e) intruder monitoring.

On the other hand, the events should be grouped in nearby regions. We use the asynchronous event clustering method described in Section III-B. It includes accuracy-efficiency trade-off mechanisms that enable easy adaptation to a given problem providing higher flexibility than existing asynchronous clustering methods. If the event camera was static, the event clustering module would suffice for detecting intruders. However, the event camera is not static and many events are originated by the static objects in the scenario. We adopt the term of *Attention Priority Map* (APM) from neuroscience research [31] for a module that captures the regions that triggered more events within a time frame (see Section III-C). Hence, the events originated by moving objects are assigned with higher attention priority, see Figure 2-c, than those generated by static objects in the scene. Intruders create groups of events close to corners with consistent motion in the scenario. Hence, the clustering module groups the events with high priority and the corners correctly tracked by the corner tracking module, see Figure 2-d. Finally, the proposed scheme monitors the relevant clusters (i.e the intruder) by tracking their centroids, see Figure 2-e where the intruder is marked with a green square. If a cluster is not updated in a consistent manner or contains less than a certain number of corner tracks, it is considered noisy and is filtered out.

A. Asynchronous Event-based Feature Tracking

The aim of this module is to follow active features on the scene for intruder monitoring. The corner tracking method described in [10] uses tree graphs to save information of the previous corners followed by a track. It reports remarkable accuracy. However, performing data association using all the vertices in a sub-tree and refining the tracking position is too computationally costly in our scheme with several asynchronous event processing modules. We designed a tracking method that reduces the computational needs at the cost of a slight reduction in pixel location accuracy, which is not critical in our case. Algorithm 1 shows the proposed event-based asynchronous feature tracker. Each feature $\mathbf{f} = \{t, u, v\}$ is defined by a timestamp t of the event that generated the feature and its coordinates (u, v). The tracker categorizes new features $\mathbf{f}$ as: (i) *track_update*, candidates that match with at least one of the previous tracked features; (ii) *new_track*, candidates considered as a new feature to track, and; (iii) *no_track*, discarded features. Tracked features are stored in the list $\hat{\mathbf{F}} = [\hat{\mathbf{f}}_1, \dots \hat{\mathbf{f}}_n]$, where $\hat{\mathbf{f}}_i$ is the i-th feature tracked and n is the number of tracked features.

Algorithm 1: Asynchronous event-based feature tracking

Parameters: ν
Input: $\mathbf{f}$
Output: $\hat{\mathbf{F}}$
$\tau \leftarrow$ UpdateFeatureBuffer($\mathbf{f}$,$\mathbf{B}_1$,$\mathbf{S}$)
if *ActiveFeatures*($\mathbf{f}$,$\mathbf{S}$) $> \nu$ **then**
  $\mathbf{M} \leftarrow$ SearchCandidates($\hat{\mathbf{F}}$,A,$\mathbf{f}$) ▷ Find matches in $\hat{\mathbf{F}}$.
  if $\mathbf{M} \neq \emptyset$ **then**
    $\hat{\mathbf{F}} \leftarrow$ TrackUpdate($\mathbf{f}$) ▷ Update $\hat{\mathbf{F}}$ by $\mathbf{f}$.
  else
    $\hat{\mathbf{F}} \leftarrow$ NewTrack($\mathbf{f}$) ▷ Add a new feature to $\hat{\mathbf{F}}$
  end
end
$\hat{\mathbf{F}} \leftarrow$CleanTracker($\mathbf{M}$, τ) ▷ Remove old features

The candidate evaluation consists of analyzing the occurrence of previous tracks $\hat{\mathbf{f}}_i$ in a neighborhood area A around the candidate $\mathbf{f}$. Previously tracked features within the neighbourhood area s.t. $\hat{\mathbf{f}}_i \in A$ are appended to the list of feature matches $\mathbf{M}$. The candidate is categorized as *track_update* when $\mathbf{M} \neq \emptyset$. In this case, the tracked feature $\hat{\mathbf{f}}_i$ in $\mathbf{M}$ with the smallest index i (i.e the oldest track) is updated by the candidate $\mathbf{f}$ in the list of tracked features $\hat{\mathbf{F}}$. Otherwise, the candidate is categorized as *new_track* by adding $\mathbf{f}$ to $\hat{\mathbf{F}}$. Further, features from noisy events are categorized as *no_track*. Noise rejection is performed by discarding candidates with a number of previous features around $\mathbf{f}$ lower than a threshold ν using $\mathbf{S}$, a spatio-temporal distribution of the previous features. Thus, only features with continuous occurrence are tracked. Non-noisy candidates are assumed to have $\nu \geq 2$ previous features in its boundary.

Previously tracked features are removed from $\hat{\mathbf{F}}$ in two cases. First, when a candidate $\mathbf{f}$ matches more than one element of $\hat{\mathbf{F}}$, only the oldest tracked feature $\hat{\mathbf{f}}_i$ in $\mathbf{M}$ is updated and the remaining elements on the list are deleted

Preprint version of:
Rodríguez-Gómez, J. P., Gómez Eguíluz, A., Martínez-de Dios, J. R., Ollero, A. (2020, June).
Asynchronous event-based clustering and tracking for intrusion monitoring in UAS.
In IEEE International Conference on Robotics and Automation (ICRA 2020)

from $\hat{\mathbf{F}}$. Second, tracked features $\hat{\boldsymbol{f}}_i$ with a timestamp lower than the dynamic time reference τ are removed from $\hat{\mathbf{F}}$. Hence, the algorithm cancels the tracks that are not updated according to the dynamics of the scene. The dynamic time reference τ is obtained by retrieving the oldest timestamp from a fixed size buffer $\mathbf{B}_1$ containing the last m features. This time-based forgetting horizon is preferred instead of a fixed value since a fixed temporal interval is affected by the velocity of the camera, as reported in [32]. $\mathbf{S}$ is the 2D representation of the locations of the features in $\mathbf{B}_1$. Hence, an under-estimation of τ would entail an incomplete representation while, an over-estimation, would originate motion blur in $\mathbf{S}$, as reported in [33].

B. Event Asynchronous Clustering for Scene Segmentation

The main role of this method within the proposed scheme is to perform object segmentation in an event-cluttered scenario. To exploit the sequential nature of AER data, a method capable of grouping the events in an asynchronous manner is required. Events are generally triggered at the contour of the objects and, therefore, they are often distant from the cluster centroid. The method described in [26] is an asynchronous adaptation of mean-shift clustering that uses a Gaussian kernel centered at the cluster centroid to decide if events are assigned to the cluster. This approach works well with clearly isolated objects but it is not designed for event-cluttered unstructured scenarios such as in our problem. Our method finds clusters of events with spatial continuity within a dynamic time frame by analyzing the proximity of each new event to a random sample of the events already assigned to the cluster. The method shows high robustness in event-cluttered scenarios, it does not require a-priori knowledge and adapts to an arbitrary number of objects in the scene.

In a nutshell, our method evaluates the spatio-temporal proximity of each new event to the existing clusters. If the event is close to only one cluster, it is assigned to it and the cluster is updated. If the event is close to more than one cluster, the clusters are merged into one and the new cluster is updated. If the event is not close to any existing cluster, a new cluster is created. The method is shown in Alg. 2. Similarly to the tracker in Section III-A, the method keeps a buffer $\mathbf{B}_2$ with the m more recent events. The buffer is updated with each new event. τ is the timestamp of the oldest event in the buffer. τ is used as a time horizon that adapts to the camera motion and scene dynamics.

A cluster c is defined by: its centroid $\boldsymbol{\mu}_c$, a weighted average $\hat{\boldsymbol{\mu}}_c$ and a list $\boldsymbol{\Theta}_c$ of the events with a timestamp greater than τ that were assigned to the cluster. The list $\boldsymbol{\Theta}_c$ is kept updated by removing the events with timestamp lower than τ, so that each cluster can keep an updated representation of a moving object. A cluster is removed when all its assigned events are older than τ. When a cluster c is updated with event $\mathbf{e}$, its running weighted average $\hat{\boldsymbol{\mu}}_c$ is computed as:

$$\hat{\boldsymbol{\mu}}_c = \big(\alpha\mathbf{x} + (1-\alpha)\hat{\boldsymbol{\mu}}_c\big)/2, \tag{1}$$

where $\alpha \in [0,1]$ is the weight parameter. We empirically found that weighting the average towards the new event (i.e. $\alpha \gg 0.5$) improves the clustering performance since the events of an object are consistently triggered at specific parts of its contour during shorts period of time. We used $\alpha = 0.9$ in the performed experiments.

Algorithm 2: Asynchronous event clustering algorithm

Parameters: r, κ
Input: $\mathbf{e}$
Output: $\boldsymbol{\mu}$
$\tau \leftarrow$ UpdateEventBuffer($\mathbf{e}$,$\mathbf{B}_2$)
UpdateClusters(τ); ▷ Delete old events.
$\mathbf{L} \leftarrow$ EvaluateClusterProximity($\mathbf{e}$,r,κ)
if $\boldsymbol{L} = \emptyset$ **then**
  CreateNewCluster($\mathbf{e}$)
else if *Size(**L**)* $= 1$ **then**
  AddToCluster($\mathbf{e}$,$\mathbf{L}$) ▷ Add event to cluster.
else
  MergeClusters($\mathbf{e}$, $\mathbf{L}$) ▷ Merge clusters & add event.
end
$\boldsymbol{\mu} \leftarrow$ ComputeClusterCentroid($\mathbf{L}$)

The Manhattan distance between coordinates of the new event $\mathbf{e}$ and $\hat{\boldsymbol{\mu}}_c$ is used to evaluate event-cluster proximity. Additionally, we evaluate the proximity between the event and κ random samples drew from $\boldsymbol{\Theta}_c$, i.e. the list of events assigned to that cluster. Event $\mathbf{e}$ is assigned to cluster c if any of these distances is below a given threshold r (typically $r = 10$). All clusters are evaluated and the clusters assigned to $\mathbf{e}$ are included in list $\mathbf{L}$. If $\mathbf{L}$ contains one only element, the cluster is updated adding $\mathbf{e}$ to $\boldsymbol{\Theta}_c$ and using Eq. (1). If $\mathbf{L}$ contains more than one element, all clusters in $\mathbf{L}$ are merged into one and the resulting cluster is updated with $\mathbf{e}$. An event with $\mathbf{L} = \emptyset$ creates a new cluster.

The value of κ, number of samples in the cluster contour used to evaluate event-cluster proximity, entails a trade-off between computational cost and clustering accuracy. In our experiments, $\kappa = 100$ provided a good accuracy with low computational cost, suitable for online execution.

C. Event-based Attention Priority Map

Existing event-based surveillance methods rely on static cameras [12] [13]. Their implementation in moving robots is limited: when the event camera moves, many events are originated by static objects in the scenario, hampering the detection of actual moving intruders. The proposed approach relies on spatio-temporal information to differentiate the events generated as a consequence of the robot motion from those corresponding to a moving object. We assume that the regions corresponding to moving objects trigger significantly more events than static ones. We use the APM to capture those regions within a time frame by defining $\boldsymbol{\Omega} \in \mathbb{R}^2$ with a similar resolution than the event camera. The APM, $\boldsymbol{\Omega}$, is updated asynchronously with each event $\mathbf{e}$ by increasing the values in $\boldsymbol{\Omega}$ in a l-sized window centered at coordinate $\mathbf{x} = (u, v)$ of event $\mathbf{e}$ as follows:

$$\boldsymbol{\Omega}_{ij} = \boldsymbol{\Omega}_{ij} + l - D(\mathbf{x}, \mathbf{y}) + 1, \tag{2}$$

Preprint version of:
Rodríguez-Gómez, J. P., Gómez Eguíluz, A., Martínez-de Dios, J. R., Ollero, A. (2020, June).
Asynchronous event-based clustering and tracking for intrusion monitoring in UAS.
In IEEE International Conference on Robotics and Automation (ICRA 2020)

where $D(\cdot)$ is the Manhattan distance, $\mathbf{y} = (i, j)$, $i \in [u - \frac{l-1}{2}, u + \frac{l-1}{2}]$, and $j \in [v - \frac{l-1}{2}, v + \frac{l-1}{2}]$. Similarly, events older than τ are forgotten, i.e. their values are removed from $\mathbf{\Omega}$. Hence, the values of $\mathbf{\Omega}$ represent the number of events triggered within a region during a dynamic time frame. The values in $\mathbf{\Omega}$ are normalized in the range $[0, 1]$. An event at coordinate $\mathbf{x}$ is considered to attract attention when $\mathbf{\Omega}_{u,v}$ is greater or equal than a threshold $\omega \in [0, 1]$. Thus, only events with high attention priority are considered for clustering.

In order to improve the computational cost, the scheme includes a mechanism to reduce the number of events that are processed by APM. The event stream input to APM is randomly sampled and only a percentage $\gamma \in [0, 1]$ of events were processed. Although an accuracy/computational cost trade-off exists, event cameras onboard UAS generate so many events that processing only a faction of them in APM is a good choice to reduce the computational cost with no significant influence on performance. This mechanism is experimentally validated Section IV-B.

IV. Experimental Results

The proposed scheme has been validated in sets of experiments in highly complex and unstructured scenarios for different conditions including day/night operation and with one/several intruders. The UAS used was a DJI Flamewheel F550 Drone shown in Figure 2-top. The UAS was equipped with a DAVIS 346 pointing at a pitch angle of -45 degrees and an INTEL® NUC6i7KYK2 for logging and online computation. The total weight of the platform was 3.49 Kg. The described event-based scheme was implemented in ROS Kinetic. The following parameters were selected for all the experiments performed: $m = 100$, $\kappa = 100$ and $\omega = 0.5$.

The proposed feature tracking and event clustering methods were also validated in laboratory experiments in which several laser light sources were projected on a canvas generating a continuous source of events. The events were taken as input of the clustering method and also of the tracker preventing potential feature detection failures. Different sets of experiments were performed with different number of lasers, different motion patterns and speeds. Their validation was performed manually checking the consistency of their results with the input events. In all experiments, the feature tracker followed every laser light correctly, without any tracking loss. When two or more laser lights crossed in the scene, the tracker kept the older track and assigned a new identifier to the rest of the features, as described in Section III-A. The clustering method also successfully grouped all the events generated by each laser light.

A total of 36 UAS-based outdoor experiments of the full scheme were performed. In each experiment, an intruder person moved in the surveillance area and tried to hide from the drone to simulate escape or intrusion situations. We first analyse performance in experiments performed during the day under different lighting conditions. Figure 3-left shows some intrusion monitoring results in day experiments in which the intruder tracked is marked within a green window. Our scheme processes only the DVS output of the DAVIS camera for intruder monitoring. We used the DAVIS APS output combined with the events from the DVS as ground truth for the method evaluation. The evaluation was performed counting the number of false positives, false negatives, true positives and true negatives on each of the ground truth images. With these, we computed the *Accuracy*, *Precision* and *Recall* metrics as defined in [34] to evaluate the scheme detection success rate and noise rejection. The average results obtained in all the *Daylight* experiments were *Accuracy*=0.98, *Precision*=0.99 and *Recall*=0.97, see Table I. These values demonstrate the expected good performance of the scheme.

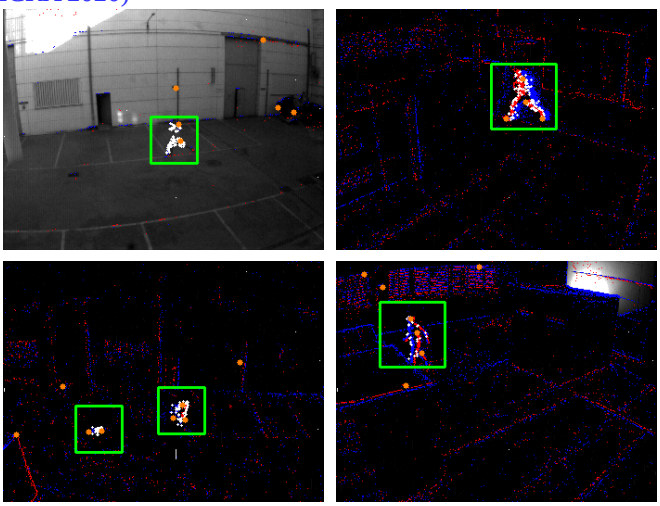


Fig. 3: Intruder monitoring during four experiments. From top-left: (i) daylight, (ii) night, (iii) multi-object, and (iv) illumination changes. Cluster events and feature tracks are represented by white and orange pixels respectively.

A. Robustness Analysis

The proposed scheme has to be robust to the different conditions that can be found in typical surveillance applications. We evaluated its performance in sets of experiments with increasing degree of difficulty: a) experiments performed during the *night* –where many intrusion events might happen, b) *multi-track* experiments performed during the night with several intruders in the scene –human intruder and a drone intruder, and c) *dynamic lighting* experiments performed in the night with strong temporal lighting changes caused by either moving or flashing lights in the scene.

Figure 3 shows the results obtained in different sets of experiments performed under these conditions: *daylight*, *night*, *multi-track* and *dynamic lighting*. The average performance

TABLE I: Performance results by processing the complete set of generated events.

Experiment	Precision	Recall	Accuracy
Daylight	0.99	0.97	0.98
Night	0.97	0.96	0.97
Multi-track	0.96	0.96	0.95
Dynamic lighting	0.97	0.88	0.91

Preprint version of:
Rodríguez-Gómez, J. P., Gómez Eguíluz, A., Martínez-de Dios, J. R., Ollero, A. (2020, June).
Asynchronous event-based clustering and tracking for intrusion monitoring in UAS.
In IEEE International Conference on Robotics and Automation (ICRA 2020)

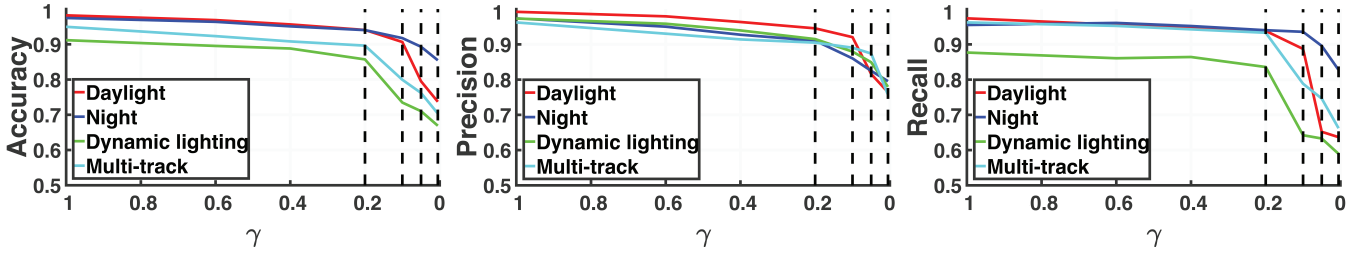


Fig. 4: Performance results for all experiments. *Accuracy*, *Precision* and *Recall* as a function of γ.

metrics in *Accuracy*, *Precision* and *Recall* are shown in Table I. The best performance was obtained in *daylight* experiments. In *night* experiments, the lower signal-to-noise ratio produced by the low illumination conditions originated a slight performance degradation w.r.t. *daylight* of 1-2%.

In the *multi-track* experiments, having several intruders moving in the scene could impact on the APM module when one of them moves significantly faster than the rest. Although we used two different types of intruders, their speed difference was not enough and the APM performed as expected. The very low difference in *Accuracy* and *Precision* was caused by false positives originated from noise. The *dynamic lighting* experiments were the most challenging. They were performed during the night, and the pointing of some light sources was moved in the scenario. Lighting changes originate noisy events all around the scene that hampered intruder monitoring as it is shown by the *Recall* value. Despite these effects, the performance results were still good and show the robustness of the proposed scheme under rather challenging lighting conditions.

B. Real-time Feasibility

Asynchronous processing schemes fully exploit the sequential nature of the event stream but are more computationally demanding since they require processing event-by-event. The proposed scheme was carefully designed to enable its on-line on-board execution. One relevant mechanism adopted to reduce the computational cost is the sampling of input events to the APM module. Parameter $\gamma \in [0, 1]$ is the percentage of the input events that are processed in APM and hence, also in event clustering. γ should be set to adjust the computational cost of the scheme for running on real time adapting to the scenario particularities and the hardware onboard the UAS. The results reported in Table I were obtained with $\gamma = 1.0$.

Figure 4 shows the values obtained for *Accuracy*, *Precision* and *Recall* using different values of γ for each type of experiments. In order to cancel the randomness of the γ sampling, each experiment was repeated 10 times and the results were averaged. Figure 4 shows the average metrics obtained with all the experiments of each type. All performance metrics tend to degrade due to the reduction in the number of events processed by the event clustering module. However, the performance decay is low and smooth in all types of experiments for $\gamma \geq 0.2$. These results validate the use of γ to largely reduce the computational cost without significant degradation. When $\gamma = 0$ the event clustering module is only fed with the tracked corners from both moving and static objects, which increases the number of false positives, degrading the performance.

In all above experiments using $\gamma = 0.2$ the proposed scheme processed each event in an average of $5.72\mu s$ in the UAS onboard hardware. In average the camera generated 115,000 events per second, i.e. all the events generated in one second were processed in $0.66s$. Each *daylight*, *night* and *multi-track* experiment was executed in real time along the full experiment: the scheme was capable of processing the event stream using 20% of the events for event clustering while discarding the rest without significant performance degradation. The same happened during the greater part of the *dynamic changes* experiments, except the few occasions in which a light source pointed directly to the event camera. In these cases, the camera produced >175,000 events in one second and the overloading events were discarded without significant overall performance degradation.

V. Conclusions and Future Work

This paper proposes a scheme for intrusion monitoring using UAS equipped with event cameras. As event cameras provide high dynamic range, our scheme is robust against challenging illumination conditions. While previous works either relied on *event images* or traditional cameras, this paper presents an asynchronous event-based method for intrusion monitoring that is resilient to motion blur by performing event-by-event processing. The proposed method was evaluated onboard a UAS equipped with a DAVIS346 sensor. The results showed significant robustness at monitoring intrusions in real-time with high accuracy for a number of challenging scenarios.

This work has been developed in the context of the ERC Advanced Grant GRIFFIN project, which aims to develop flapping-wing aerial robots capable of navigating, perching and manipulating objects. The fast response of event cameras provide the necessary capabilities for flapping-wing robots (i.e. ornithopters), which perform faster than multirotors and suffer from higher levels of mechanical vibration. Our future work is to adapt and extend the proposed scheme to be used onboard an ornithopter robot.

References

[1] S. Berrahal, J.-H. Kim, S. Rekhis, N. Boudriga, D. Wilkins, and J. Acevedo, "Border surveillance monitoring using quadcopter uav-

Preprint version of:
Rodríguez-Gómez, J. P., Gómez Eguíluz, A., Martínez-de Dios, J. R., Ollero, A. (2020, June).
Asynchronous event-based clustering and tracking for intrusion monitoring in UAS.
In IEEE International Conference on Robotics and Automation (ICRA 2020)

aided wireless sensor networks," *Journal of Communications Software and Systems*, vol. 12, no. 1, pp. 67–82, 2016.

[2] S. Rasmussen, K. Kalyanam, and D. Kingston, "Field experiment of a fully autonomous multiple uav/ugs intruder detection and monitoring system," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2016, pp. 1293–1302.

[3] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, "Are we ready for autonomous drone racing? the uzhfpv drone racing dataset," in *IEEE Int. Conf. Robot. Autom.(ICRA)*, 2019.

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

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

[6] A. Gómez Eguíluz, J. P. Rodríguez-Gómez, J. Paneque, P. Grau, J. R. Martinez De-Dios, and A. Ollero, "Towards flapping wing robot visual perception: Opportunities and challenges," in *International Workshop on Research, Education and Development on Unmanned Aerial Systems (RED-UAS)*, 2019.

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

[8] N. J. Sanket, C. M. Parameshwara, C. D. Singh, A. V. Kuruttukulam, C. Fermuller, D. Scaramuzza, and Y. Aloimonos, "Evdodge: Embodied ai for high-speed dodging on a quadrotor using event cameras," 2019.

[9] 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)*. IEEE, 2019.

[10] I. Alzugaray and M. Chli, "Ace: An efficient asynchronous corner tracker for event cameras," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 653–661.

[11] V. Vasco, A. Glover, E. Mueggler, D. Scaramuzza, L. Natale, and C. Bartolozzi, "Independent motion detection with event-driven cameras," in *2017 18th International Conference on Advanced Robotics (ICAR)*. IEEE, 2017, pp. 530–536.

[12] 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)*. IEEE, 2016, pp. 1–9.

[13] Z. Jiang, X. Pengfei, 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.

[14] 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.

[15] T. Stoffregen and L. Kleeman, "Simultaneous optical flow and segmentation (sofas) using dynamic vision sensor," *arXiv preprint arXiv:1805.12326*, 2018.

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

[17] A. Mitrokhin, C. Ye, C. Fermuller, Y. Aloimonos, and T. Delbruck, "Ev-imo: Motion segmentation dataset and learning pipeline for event cameras," *arXiv preprint arXiv:1903.07520*, 2019.

[18] A. Glover and C. Bartolozzi, "Event-driven ball detection and gaze fixation in clutter," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 2203–2208.

[19] ——, "Robust visual tracking with a freely-moving event camera," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3769–3776.

[20] 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)*. IEEE, 2018, pp. 1–9.

[21] 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.

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

[23] E. Mueggler, C. Bartolozzi, and D. Scaramuzza, "Fast event-based corner detection." in *BMVC*, 2017.

[24] 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.

[25] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-based feature tracking with probabilistic data association," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4465–4470.

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

[27] E. Mueggler, B. Huber, and D. Scaramuzza, "Event-based, 6-dof pose tracking for high-speed maneuvers," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2761–2768.

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

[29] E. Mueggler, C. Bartolozzi, and D. Scaramuzza, "Fast event-based corner detection." in *BMVC*, 2017.

[30] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-based feature tracking with probabilistic data association," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4465–4470.

[31] C. Mo, D. He, and F. Fang, "Attention priority map of face images in human early visual cortex," *Journal of Neuroscience*, vol. 38, no. 1, pp. 149–157, 2018.

[32] 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.

[33] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler, "Online multi-target tracking using recurrent neural networks," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[34] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 233–240.