This paper has been accepted for publication at IEEE International Conference on Robotics and Automation (ICRA) Workshop on Unconventional Sensors in Robotics, Paris, 2020.

ASAP: Adaptive Scheme for Asynchronous Processing of event-based vision algorithms

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

***Abstract*— Event cameras can capture pixel-level illumination changes with very high temporal resolution and dynamic range. They have received increasing research interest due to their robustness to lighting conditions and motion blur. Two main approaches exist in the literature to feed the event-based processing algorithms: packaging the triggered events in *event packages* and sending them one-by-one as *single events*. These approaches suffer limitations from either processing overflow or lack of responsivity. Processing overflow is caused by high event generation rates when the algorithm cannot process all the events in real-time. Conversely, lack of responsivity happens in cases of low event generation rates when the event packages are sent at too low frequencies. This paper presents *ASAP*, an adaptive scheme to manage the event stream through variable-size packages that accommodate to the event package processing times. The experimental results show that *ASAP* is capable of feeding an asynchronous event-by-event clustering algorithm in a responsive and efficient manner and at the same time prevents overflow.**

***Index Terms*— event camera, aerial robots, perception systems, event-based processing, asynchronous computation**

I. Introduction

The advent of event-based cameras has motivated increasing research interest in their application to robotics. Event cameras are neuromorphic sensors that capture the asynchronous illumination changes at pixel level and μs resolution. They provide high dynamic range and are insensitive to motion blur. Additionally, they are lightweight and have low power consumption. A good number of successful techniques have been proposed in the last years evidencing their capabilities [1].

The current trend in robotics is to group the received events in frames, i.e. *event images*. Processing of *event images* enables the design of complex and elaborated techniques, similar to those from traditional computer vision. However, that approach does not always fully exploit the sequential and asynchronous capabilities of the data stream provided by event cameras. For instance, *event images* can create motion blur in some cases and, some works, e.g. work [2] implemented specific mechanisms to mitigate it. Asynchronous *event-by-event* processing usually has higher computational needs [3], [4] and is sometimes combined with *event images* for high-level processing, see e.g. [5].

This work was supported by the European Research Council as part of GRIFFIN ERC Advanced Grant 2017 (Action 788247), AERIAL-CORE project (H2020-2019-871479), 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:rautaplop@alum.us.es, {ageguiluz, jdedios, aollero}@us.es

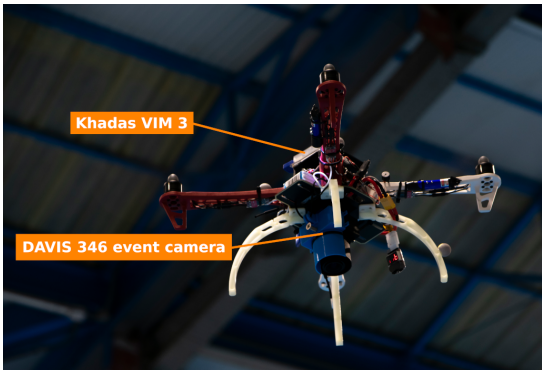

Fig. 1: Aerial robot based on *DJI Flamewheel F450* equipped with a *DAVIS 346* event camera and a low-cost *Khadas VIM3* used for on-board computation.

Two main approaches exist in the literature to feed the event-based algorithms: using *event packages* and *single events*. Most of the existing frameworks, such as YARP [6] and Dynamic Vision System [7], rely on packaging the triggered events and transmitting them as output of the camera in packages. While these mechanisms prevent the algorithms' communication overhead, they are not efficient when the algorithm is capable of processing the events in one package before the next package is received. Differently, the work in [8] presented a framework for *single-event* handling that consists of three modules: an I/O library, a computation toolbox, and a visualization tool. Although event buffers were still used for camera communication, their framework suppresses the use of buffers between the tools. Although *single-event* delivery is efficient when the algorithm can process the events faster than they are received, feeding the algorithm at a temporal resolution of μs can result in the computational overflowing.

Therefore, there is a trade-off between responsiveness and the risk of system overflow when feeding the event-based algorithms using *single events* or *event packages*. While static scenes generate a low number of events per second and allow the use of some *event-by-event* algorithms, dynamic and complex scenes often generate huge amounts of events in short periods. Additionally, event perception is largely influenced by the movement of the robot [9]. Therefore, real-time computation on-board a robot might not be possible for certain scenarios and hardware specifications. such as that used on the aerial robot shown in Fig. 1, which is the robot that has performed the experiments reported in this paper. In particular, the processing capability of aerial robots

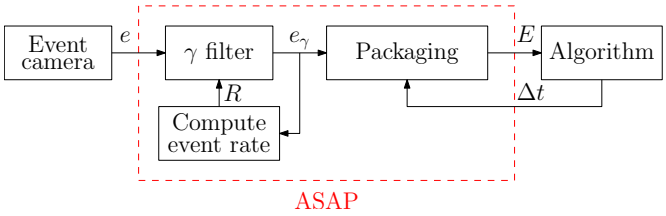

Fig. 2: Simplified scheme of *ASAP* for adaptive event packaging.

is limited mainly by their payload. In consequence, UAVs are often equipped with on-board computers with moderate computational resources, such as that installed on the aerial robot used in the reported experiments, which is shown in Fig. 1. This issue is even more relevant in flapping-wing robots since their payload limitations are more severe than in multi-rotors [10].

This work presents *ASAP*, a scheme to adapt the event packaging such that an asynchronous *event-by-event* algorithm can process the events as soon as possible without overflowing.

II. Method Description

The proposed method relies on two main mechanisms. First, the size of the *event packages* is chosen according to the time required by the algorithm to process the last packages. Second, when the hardware limitations preclude the algorithm to process all events (i.e. for a maximum package size), some events are discarded in order to avoid overflowing. Our work [11] explored the effect of discarding events in a random manner to reduce computational cost and showed that the method could work using only 20% of the full event stream without significantly affecting the algorithm performance. *ASAP* adopts this approach and makes use of a random event discard procedure to reduce the risk of overflowing an even-by-event processing algorithm.

Figure 2 shows the proposed adaptive event packaging scheme. The first mechanism makes use of a close-loop approach to adapt the size of the event packages by considering time devoted by the asynchronous event-by-event algorithm to process the event package. A closed-loop mechanism established between modules *Packaging* and *Algorithm* adapts the size of the event packages according to the time required by the algorithm for processing the previous event package. The objective of this mechanism is to synchronize (i.e. make equal) the time required by the algorithm to process the events contained in the package and the temporal difference between the newest and oldest events in that package.

The second mechanism is designed for cases in which the rate of the triggered event stream is too high to be processed in real-time by the event-by-event processing algorithm. Relying on the findings in [11], this mechanism (i.e. γ *filter* module) performs a random event discard procedure to reduce the risk of algorithm overflowing. R is the rate of events after event filtering by this module. The filtering rate γ is dynamically adapted such that the filtered event rate R is within a range that the algorithm can process in a responsive manner. If R is higher than an upper bound a, the γ *filter* increases the number of discarded events using a simple adaptive criterion. a was selected experimentally analyzing three types of scenarios with different event rates: low motion –low event rate– moderate motion and high motion scenarios. a was selected as an average event rate value suitable for most scenarios.

Fig. 3: Experimental scenarios. Left) Aerial robot used for the UAV on-board evaluation experiments. Right) Pattern used for the event overflow experiments.

III. Experimental Results

This section presents the experimental evaluation of *ASAP* when processing the events with the asynchronous event-based clustering method presented in [11]. Two experiments were performed: highly abrupt camera motion and UAV on-board evaluation. The first experiment was performed manually using a *DAVIS346* event camera oriented such that the pattern in Fig. 3-right was in the camera field of view during the whole experiment. The vibration speed of the camera was increased progressively during 5 s. Figure 4 shows the number of events per package, the value of R and the value of γ along the experiment. We selected $a = 5 \cdot 10^6$. With low vibration level, packages included very few events. As vibration level increased, more and more events were transmitted in each package using $\gamma = 1$. When the event rate exceeded the boundary $a = 5 \cdot 10^6$, γ was reduced to balance the filtered event rate (R) with the time required

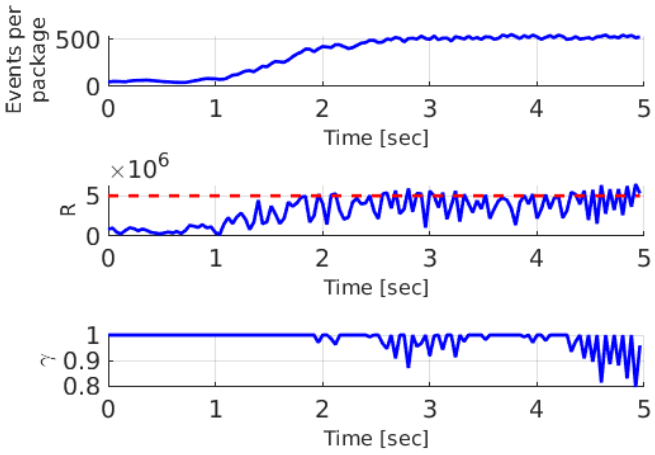

Fig. 4: Results in highly abrupt motion experiments.

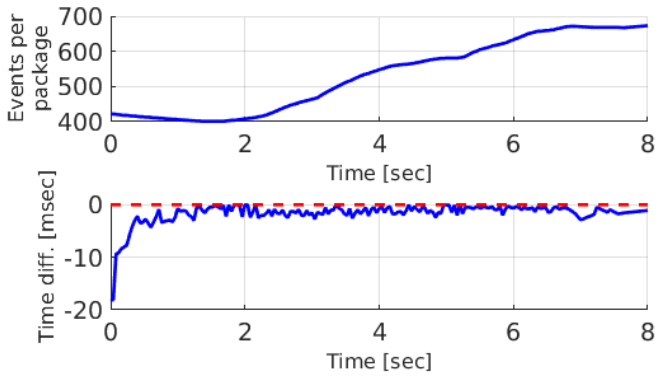

Fig. 5: Experimental results in UAV on-board evaluation.

by the algorithm to process the event package running on the specific hardware, and thus adapting to the particular implementation.

Additionally, the performance of *ASAP* was evaluated on-board an UAV in an indoor scenario as shown in Figure 3-left. The experimental setup consists of a *DAVIS346* event camera installed on-board a *DJI Flamewheel F450* frame with a *PixRacer* autopilot (see Fig. 1). A low-cost *Khadas VIM3* board is used for real-time running the event-based clustering method described in [11] and for logging the results. *ASAP* was implemented in C++ on top of the UAL abstraction layer [12] using *ROS Kinetic* and the *PX4* low-level controller. In the experiment, the UAV flew over several chairs as shown in Fig. 3-left. The number of chairs increased as the UAV moved towards the goal and hence, the number of events generated over time also increased (see Fig. 5-top). However, the event rate never exceeded the threshold a and, therefore, the full event stream (i.e. $\gamma = 1$, no events were discarded) was packaged –and also on-line processed by the clustering algorithm– during the whole experiment.

Moreover, Fig. 5-bottom shows the difference between the time required by the clustering algorithm described in [11] to process a package and the time difference between the newest and oldest events in that package. As long as the time difference remains negative, real-time processing is guaranteed as the algorithm will always process a package before the next one is received. Thus, *ASAP* is capable of adapting the size of the packages to keep the time difference negative –preventing the algorithm from overflowing– and near zero –synchronizing algorithm executing and event packaging, which ensures responsive event processing.

IV. Conclusions and Future Work

This paper presents *ASAP*, an adaptive scheme for dynamic event packaging that enables responsive event processing while preventing processing overflow. The experimental results show that the proposed approach regulates the event stream fed to an *event-by-event* clustering algorithm and it is capable of filtering upon saturation. Our future work will focus on enhancing the scheme response to abrupt changes in the event stream and on evaluating the scheme in a wider range of asynchronous event-based algorithms.

References

[1] 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.

[2] N. J. Sanket, C. M. Parameshwara, C. D. Singh, A. V. Kuruttukulam, C. Fermüller, D. Scaramuzza, and Y. Aloimonos, "Evdodge: Embodied ai for high-speed dodging on a quadrotor using event cameras," *arXiv preprint arXiv:1906.02919*, 2019.

[3] 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.

[4] 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.

[5] 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.

[6] A. Glover, V. Vasco, M. Iacono, and C. Bartolozzi, "The event-driven Software Library for YARP — With Algorithms and iCub Applications," *Frontiers in Robotics and AI*, vol. 4, p. 73, 2018.

[7] 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*, 2014, pp. 2761–2768.

[8] A. Marcireau, S. H. Ieng, and R. B. Benosman, "Sepia, tarsier and chameleon: a modular c++ framework for event-based computer vision." *Frontiers in Neuroscience*, vol. 13, p. 1338, 2019.

[9] A. Pequeño-Zurro, D. Shaikh, and I. Rañó, "Temporal changes in stimulus perception improve bio-inspired source seeking," *arXiv preprint arXiv:1903.10279*, 2019.

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

[11] J. P. Rodríguez-Gómez, A. Gómez Eguíluz, J. R. Martínez-de Dios, and A. Ollero, "Asynchronous event-based clustering and tracking for intrusion monitoring in uas," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020.

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