Towards a Dynamic Vision Sensor-based Insect Camera Trap

Eike Gebauer*[1,2], Sebastian Thiele*[1,2], Pierre Ouvrard[3], Adrien Sicard[3], Benjamin Risse[1,2]
[1]Institute for Geoinformatics, University of Münster, Germany
[2]Faculty of Mathematics and Computer Science, University of Münster, Germany
[3]Dep. of Plant Biology, Uppsala BioCenter, Swedish University of Agricultural Sciences, Sweden
firstname.lastname@uni-muenster.de[1], firstname.lastname@slu.se[3]

Abstract

This paper introduces a visual real-time insect monitoring approach capable of detecting and tracking tiny and fast-moving objects in cluttered wildlife conditions using an RGB-DVS stereo-camera system. By building on the intrinsic benefits of event vision data acquisition, we demonstrate that insect presence can be detected at an extremely high temporal rate (on average more than 40 *times real-time) while surpassing the spatial and spectral sensitivity of conventional colour-based sensing. Our DVS-based detection and tracking algorithm extracts insect locations over time, and we evaluated our system based on* 81104 *manually annotated stereo-frames with* 34453 *insect appearances featuring highly varying scenes and imaging conditions (including clutter, wind-induced motion, etc.). Comparing our algorithm to two state-of-the-art deep learning algorithms reveals superior results in both detection performance and computational speed. Using the DVS as a trigger for the temporally synchronised RGB camera, we are able to correctly identify* 73% *of images with and without insects which can be increased to* 76% *with parameters optimised for different scenes. Overall, our study suggests that DVS-based sensing can be used for visual insect monitoring by enabling reliable real-time insect detection in wildlife conditions while significantly reducing the necessity for data storage, manual labour and energy.*

1. Introduction

Insects are by far the largest and most diverse class of animals and are pivotal for the ecosystem while having a tremendous impact on food production, health and the economy [1, 6, 28]. During the last decade, an alarming decline in insect biomass, abundance and diversity has been reported [34, 36], which has resulted in an increased demand for insect monitoring strategies [10]. Apart from manual censuses using conventional techniques such as Malaise traps [23] image-based monitoring systems appear to be the most promising solution for high-throughput detection and classification of insects at scale [3].

From a computer vision point of view, quantifying the abundance and behaviour of insects in natural environments is however notoriously more difficult than recording bigger vertebrates for which dedicated camera trap systems can be used to detect motion in the field of view of the vision sensor [13]. Due to the small size, potentially low contrast, and relatively fast movement speeds in front of cluttered and dynamic backgrounds, motion detectors such as passive infrared sensors are not sufficient to detect and track insects reliably [15, 16]. As a consequence, time-lapse and high frame-rate monitoring systems have been introduced to collect continuous data independent of the presence of insects, which, however, result in a huge amount of non-informative images (i.e. without insects present), high energy consumption and the need for real-time image processing on limited computing hardware.

Related Work To address these challenges, a variety of insect monitoring systems using dedicated hardware and software modules have been introduced in the past. For example, Naqvi *et al*. demonstrated that scheduled frame capturing outperforms motion-activated imaging using off-the-shelf camera trap hardware [27]. Others have built custom off-grid systems either based on ultra-low power hardware and offline processing [11] or on edge compute units enabling on-board processing at increased energy consumption [7].

If energy and storage limitations are neglected, continuous recordings can be processed in an offline fashion. For example, Ratnayake *et al*. combined kNN background subtraction with YOLOv2 to detect honeybees during foraging [30], which has later been extended towards additional flower detection using YOLOv4 [29]. From a technical point of view, the central challenge is to detect tiny low-contrast objects in cluttered environments.

*These authors contributed equally to this work.

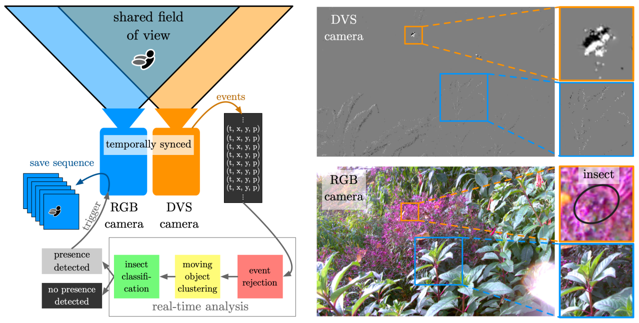

Figure 1. Overview of the DVS-RGB imaging system. Left: Continuous DVS camera events are processed using a real-time detection algorithm (see Sec. 2). If insect presence is detected in the shared field of view, the temporally synchronized RGB camera is triggered. Middle: Exemplary DVS (top) and RGB (bottom) image and zoomed crops of insect (orange) and plant motion (blue). Note that the insect can hardly be seen in the RGB image (black circle).

To address these challenges, a variety of detection strategies have been developed, including data augmentation [9], super-resolution methods [4, 21], multi-scale representation techniques [14], dedicated loss functions [22] and context-based strategies [17]. Established algorithms have also been adapted to detect small objects, including YOLOv7 [37], CenterNet [38] and CornerNet [20]. Since most of these algorithms were not designed for the particular use-case of insect detection in natural environments, they do not enable the very high frame rate and low energy consumption necessary for an effective visual insect camera trap triggered by real-time detection.

Dynamic Vision Sensors (DVS), also called event cameras, could provide a solution to these challenges. This relatively new class of neuromorphic vision sensors enables asynchronous operation paradigms in which the pixels are triggered by local brightness changes above a threshold [12]. As a result, DVS cameras output sparse events at a variable rate consisting of the (x, y)-location on the sensor array, a timestamp and the polarity (i.e. sign of the change). This paradigm results in very high temporal resolutions, low power consumption, high dynamic range and reduced data redundancy, making it highly suited for outdoor recordings [8], so that it has been used for a variety of tasks such as SLAM [18] and human pose estimation [33] to name a few examples.

Event cameras have already been used in the context of object detection and tracking. These methods are generally either based on feature detection/tracking or on clustering events over time. The former includes classical feature detection/tracking applied to integrated event images as in the work by Zhu *et al*. [39] on one side and the purely asynchronous corner tracking algorithm by Alzugaray and Chli [2] on the other. Rodríguez-Gomez *et al*. [31] later employed a purely asynchronous hybrid approach between feature tracking and event clustering by identifying corner features via eFast [25] followed by asynchronous event clustering to associate pixel regions with features.

As with feature tracking, purely cluster-based methods generally either operate on chunks of event data or asynchronously. The works by Mondal and Das [24] and Barranco *et al*. [5] fall into the former category, while Schraml and Belbachir [35] and Lagorce *et al*. [19] proposed asynchronous methods which operate on successive events. Schraml and Belbachir assign events to existing clusters based on proximity, cluster radius and cluster activity and create new clusters dynamically. In contrast, Lagorce *et al*. introduced a framework for event clustering based on continuous or discrete kernels, which actively tracks a changing subset of a fixed number of clusters.

For flying drones Sanket *et al*. proposed a Deep Learning based propeller detector called EVPropNet that uses images composed of accumulated event streams in order to predict rotor locations [32]. EVPropNet employs a U-Net like architecture that was trained on simulated data and the authors demonstrate the network's performance in controlled real world scenarios.

The methods mentioned above were usually tested in static or only slightly dynamic backgrounds. In contrast, unconstrained insect habitats include highly dynamic scenes where plants, shadows and other non-insect objects induce almost constant motion cues (e.g. wind-induced motion). Moreover, the limited spatial resolution and non-optimal signal-to-noise ratios of previously available DVS sensors also limit their applicability to small insects in a relatively large field of view. Given the new generation of event cameras based on the IMX636 ES sensor (offering a 1280×720 resolution), these devices have now attained a technological level of quality, rendering insect detection and tracking possible for the first time.

Contribution Our overall goal is to present our efforts towards building the first event vision-based insect camera trap consisting of a temporally synchronised RGB-DVS stereo camera system. We demonstrate that the core benefits of DVS, namely very high temporal resolutions with high dynamic range, can be used in a probabilistic detection and tracking algorithm to identify insects within the field of view of the cameras. Given our highly optimised algorithm, insect detections can be computed with, on average, 4800% of the necessary real-time performance and are therefore fast enough to trigger the RGB camera resulting in lower power consumption and reduced data storage requirements. To evaluate our system, we generated a dedicated dataset consisting of six different wildlife imaging scenarios recorded in the Botanical Garden of the University of Münster with varying background and foreground conditions, including strong clutter, wind-induced motion, moving shadows and other disturbing artefacts. In total, we annotated 41897 insect detections in 81104 frames, and our algorithm was able to correctly identify 81.02% of all frames with insect presence while providing a detection F1 score ranging from 69.62% for suitable imaging conditions to 22.35% for inappropriate situations. We compare our algorithm with YOLOv7 (as a baseline for state-of-the-art object detection [37]) and EVPropNet (as a baseline for event-based flying object detection [32]) and demonstrate that our algorithm outperforms both approaches in detection accuracy and computational performance. Our results suggest that DVS-based sensing can be advantageous compared to conventional imaging hardware to detect small insects in cluttered and dynamic environments. At the time of writing this article, event cameras are still relatively expensive compared to consumer RGB cameras. However, increasing demand could lead to cheaper event based systems in the future. Code and dataset are available at `https://zivgitlab.uni-muenster.de/cvmls/neuromorphic-insect-detection`.

2. Methods

2.1. Hardware System

The imaging system uses a full frame RGB camera (Basler acA1920-150uc 2.3 MP 150fps; lens: Kowa LM6JC 6mm/F1.4 C-Mount) and an event-based camera (Prophesee EVK-3 HD 0.92 MP; lens: Kowa LM5NCL 4.5mm/F1.4 C-Mount), which were arranged in an axis-parallel stereo camera system with a minimum baseline of 60 mm to provide an appropriate shared field-of-view (cf. Fig. 1). The lenses were chosen to achieve a similar opening angle at a working distance of 1.5m. Both cameras attached to a tripod via a common base plate can then be positioned in front of or above plants of interest.

For temporal synchronization, the `ExposureActive` signal of the full frame camera is connected to `TriggerIn` of the event-based camera. This way, a trigger event with positive polarity is injected into the event stream when the exposure of the full frame camera starts, and one with negative polarity when the exposure ends. Synchronization is achieved by matching the n-th frame after starting the full-frame camera with the n-th trigger event of the same polarity. We note that spatial stereo camera calibration can be done using existing frameworks [26]. Since we focus on using solely the DVS as a trigger for the RGB camera we ensured an overlapping field of view without explicit extrinsic camera parameter calculation.

2.2. Insect Dataset

To develop and evaluate our event-based real-time detection and tracking algorithm, we generated a benchmark dataset using the synchronized stereo-camera system described in Sec. 2.1. The DVS events were captured with its maximum 1280 × 720 resolution (without fine-tuning biases), while the frame-based camera was set to record 1920 × 1200 images at 100 fps. In total, we recorded 31 videos featuring a variety of weather, illumination and vegetation conditions and viewing angles. All recordings were made in the Botanical Garden of the University of Münster, which features a wide variety of plants and insects, making it a suitable testing ground for our system. We classified the difficulty of our scenes based on (1) the distance to the area of interest (short: dst, usually the plants/flowers in the centre of the visual field); (2) the dynamics of the area of interest (short: fd, mainly the wind-induced motion of these plants); and (3) the dynamics of the background (short: bd, motion which does not belong to the area of interest). Each category was then manually rated as low (l), medium (m) or high (h) magnitude (e.g. 'l-l-l' indicates low rating for (1)-(2)-(3)).

Six of these recordings were used to generate ground truth by accumulating the DVS event streams between each trigger signal, resulting in 81104 frames (see Fig. 4 for examples). These frames were annotated by adding bounding boxes around insects in the generated DVS images. We also added insect IDs and a binary confidence label to all bounding boxes enabling a fine-grained analysis of our algorithm. The IDs were used to identify unique insect trajectories, and the confidence label specifies the certainty of the annotator that the respective box represents an insect (high confidence) or could be an artefact (low confidence). Note that for annotations, the DVS signal was usually more convenient to use, so the RGB images were mainly used for verification purposes. In total, we identified 41897 bounding boxes (34453 confident), resulting in 29423 frames with insect presence and 174 unique trajectories. An overview of our dataset can be found in Tab. 1.

2.3. Real-Time Insect Detection & Tracking Algorithm

In order to recognize insect presence in the event stream, we developed an asynchronous detection and tracking algorithm. Our algorithm processes each event individually by (1) rejecting unwanted events; (2) detecting insect-shaped motion cues in the remaining event stream; and (3) disambiguating clusters representing flying insects from those representing moving plants and debris.

Event Rejection To enable real-time processing of spatially and temporally high-resolution event streams on low-

Scene (dst-fd-bd)	#Frames w Insects		#Frames w/o Insects		#Unique Insects		#Bboxes		Med Area (%)		Eventrate (kEv/s)
	overall	conf.	overall	conf.	overall	conf.	overall	conf.	overall	conf.	
l-l-l	8533	7206	7013	8340	26	25	9192	7850	0.200	0.214	89.095
l-h-l	2839	1617	32024	33246	5	5	2898	1617	0.102	0.167	28.317
m-h-h	6790	5913	1410	2287	56	37	15051	12479	0.061	0.065	50.751
m-m-h	4120	3213	1880	2787	22	14	4992	3920	0.043	0.052	55.647
h-l-h	396	372	5604	5628	11	10	405	380	0.005	0.004	7.981
h-h-h	6745	6450	3750	4045	54	37	9359	8207	0.022	0.022	191.479
combined	29423	24771	51681	56333	174	128	41897	34453	0.072	0.087	70.545

Table 1. Dataset overview. Scene naming scheme according to (1)-(2)-(3) with (1) the distance to the area of interest (dst); (2) the dynamics of the area of interest (fd); (3) the dynamics of the background (bd) and (1),(2),(3) $\in$ {low (l), medium (m), high (h)} magnitude (see Sec. 2.2). The number of frames with insects, without insects, unique insects and bounding boxes is given. Moreover, the median area (in %) and event rate in 1000 events per second (kEv/s) are specified.

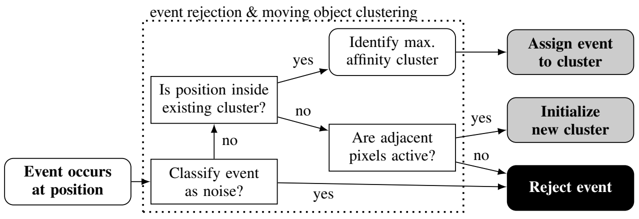
event rejection & moving object clustering
Identify max. affinity cluster
Assign event to cluster
yes
Is position inside existing cluster?
no
Initialize new cluster
yes
Are adjacent pixels active?
no
no
Event occurs at position
Classify event as noise?
yes
Reject event

Figure 2. Event rejection and object clustering. First, the event is rejected if it was generated from noise or contains redundant information. Then, existing clusters are searched for matches. If found, the best matching cluster is updated with the new event information. If no matching cluster is found and the region of pixels around the new event shows increased activity, a new cluster is created.

power hardware, noisy and redundant information have to be rejected before being passed to more complex processing steps (see Fig. 2). By combining the density matrix [35] and the "Surface of Active Events" approach [2] we derive two continuously updated attributes for each pixel (x, y), namely $\mathtt{T}_t^{x,y}$ (timestamp of the most recent event at pixel (x, y) and time t) and $\bar{\mathtt{a}}_t^{x,y}$ (average event rate at pixel (x, y) and time t exponentially weighted by recency with decay factor d):

$$\begin{aligned}
&\text{Let } E_t = \{(x,y)|\text{ event at position } (x,y) \text{ in time step t}\}\\
&T_t^{x,y} = max(\{t|(x,y) \in E_t\}) \text{ timestamp of most}\\
&\text{recent event at pixel } (x,y)\\
&\bar{\mathtt{a}}_t^{x,y} = \frac{\sum_{i=0}^{t} d^{t-i}\, \mathbb{1}_{E_t}((x,y))}{\sum_{i=0}^{t} d^i} = \frac{\mathbb{1}_{E_t}((x,y)) + d\hat{\mathtt{a}}_{t-1}^{x,y}}{1 + dw_{t-1}}\\
&\text{with } \hat{\mathtt{a}}_t^{x,y} = \sum_{i=0}^{t} d^{t-i}\, \mathbb{1}_{E_t}((x,y)) \text{ and } w_t = \sum_{i=0}^{t} d^i
\end{aligned} \tag{1}$$

These attributes are used in an event rejection algorithm summarized in Algorithm 1 which first discards events occurring too close to the previous event (i.e. redundant information [2]) and filters the remaining events based on their timestamp and pixel activity (i.e. noisy information). The rejection is controlled by three thresholds (highlighted in grey) and can be adjusted according to the recording scenario.

```
Input: Event at time $t$ and position $(x, y)$
delta $\leftarrow (t - T_t^{x,y})$;
if delta $<$ min_delta then
    return ; // Reject event as redundant information
end
$T_t^{x,y} \leftarrow t$;
if delta $>$ max_delta then
    return ; // Reject event as noise
end
update($\bar{\mathtt{a}}_t^{x,y}$);
if $\bar{\mathtt{a}}_t^{x,y} >$ noise_threshold then
    process($t, (x, y)$);
end
```
Algorithm 1: Event rejection algorithm.

The activity is defined as exponentially weighted average $\bar{\mathtt{a}}_t^{x,y}$ at time step t to smooth pixel activity over time during tracking as it can be calculated incrementally based on new data $\mathtt{a}_t$, the preceding unnormalized weighted sum $\hat{\mathtt{a}}_{t-1}$ and the preceding sum of weights w_{t-1} (see Eq. (1)). Given large gaps between events (e.g. for mostly static background pixels) we update the average by calculating a single integer power of the decay factor:

Let $(x, y) \notin E_i$ for $i \in [1, k)$

$$\begin{aligned}
\Rightarrow \bar{\mathtt{a}}_t^{x,y} &= \frac{\mathbb{1}_{E_t}((x,y)) + d^k\hat{\mathtt{a}}_{t-k}^{x,y}}{\sum_{i=0}^{k-1} d^i + d^k w_{t-k}}\\
&= \frac{\mathbb{1}_{E_t}((x,y)) + d^k\hat{\mathtt{a}}_{t-k}^{x,y}}{\dfrac{1-d^k}{1-d} + d^k w_{t-k}}
\end{aligned} \tag{2}$$

Moving object clustering Small moving objects cause increased event rates in a relatively compact area of pixels (see Figs. 1 and 3) which can be represented by bivariate normal distributions [19]. Additionally, the average rate of assigned events is exponentially weighted by recency and kept as a measure of cluster activity. Accordingly, a cluster at time t is characterized by its activity $\bar{\mathtt{a}}_t^{\mathtt{cl}}$, the centre $\mu_t^{\mathtt{cl}}$ and the covariance matrix of the underlying normal distribution $\Sigma_t^{\mathtt{cl}}$. We also compute the area of that cluster $\mathtt{A}_t^{\mathtt{cl}}$ from this distribution by extracting the number of pixels within a Mahalanobis distance (d_M) of one from the cluster centre.

When a new event passes the former event rejection step, the cluster with the highest affinity to the event position is identified. Affinity to point (x, y) is defined as

$$\mathtt{affinity} = (\mathtt{dist\_threshold} - d_M^{\mathtt{cl}}(x,y))\frac{\bar{\mathtt{a}}_t^{\mathtt{cl}}}{(\mathtt{A}_t^{\mathtt{cl}})^2}$$

with Mahalanobis distance $d_M^{\mathtt{cl}}$ from cluster. If the maximum affinity is above zero, the event is considered to be inside the area of influence of the corresponding cluster, and the event is assigned to that cluster. This way, events are assigned to clusters based on proximity while penalizing large clusters with relatively low activity.

If the event is not inside the area of influence of any existing cluster, a new cluster is created if the surrounding pixels in the activity field have a mean activity above the $\mathtt{creation\_threshold}$. Cluster initialization at time t^* is based on a neighbourhood around the event position in the activity field:

$$\bar{\mathtt{a}}_{t^*}^{\mathtt{cl}} = \sum_{(u,v)\in N^{x,y}} \bar{\mathtt{a}}_{t^*}^{u,v}$$

$$\mu_{t^*}^{\mathtt{cl}} = \sum_{(u,v)\in N^{x,y}} \frac{(u,v)^T\, \bar{\mathtt{a}}_{t^*}^{u,v}}{\bar{\mathtt{a}}_{t^*}^{\mathtt{cl}}}$$

$$\Sigma_{t^*}^{\mathtt{cl}} = \sum_{(u,v)\in N^{x,y}} \frac{((u,v)^T - \mu_{t^*}^{\mathtt{cl}})^T((u,v)^T - \mu_{t^*}^{\mathtt{cl}})\, \bar{\mathtt{a}}_{t^*}^{u,v}}{\bar{\mathtt{a}}_{t^*}^{\mathtt{cl}}}$$

with $N^{x,y}$ neighborhood around event position x, y

If an event is assigned to a cluster, the cluster activity is increased, and the distribution parameters $\mu^{\mathtt{cl}}$ and $\Sigma^{\mathtt{cl}}$ are nudged towards the new event position by weight ω:

Let $C_i \subseteq E_i$ events assigned to cluster at time i

$$\Rightarrow \bar{\mathtt{a}}_t^{\mathtt{cl}} = \frac{\sum_{i=t^*+1}^{t} d^{t-i}|C_i| + d^{t-t^*}\bar{\mathtt{a}}_{t^*}^{\mathtt{cl}}}{\sum_{i=t^*}^{t} d^{t-i}} \tag{3}$$

$$\omega = \mathtt{weight\_factor}/\,min(\bar{\mathtt{a}}_t^{\mathtt{cl}}, \mathtt{max\_weight})$$

Here $\mathtt{weight\_factor}$ provides control over the smoothing of the parameters over time, and $\mathtt{max\_weight}$ establishes a lower bound of smoothness for clusters with low activity. The decay factor d can be chosen differently to the decay factor used to calculate pixel activity. Whenever $\frac{\bar{\mathtt{a}}_t^{\mathtt{cl}}}{(\mathtt{A}_t^{\mathtt{cl}})^2}$ falls below $\mathtt{alive\_threshold}$, the cluster is removed.

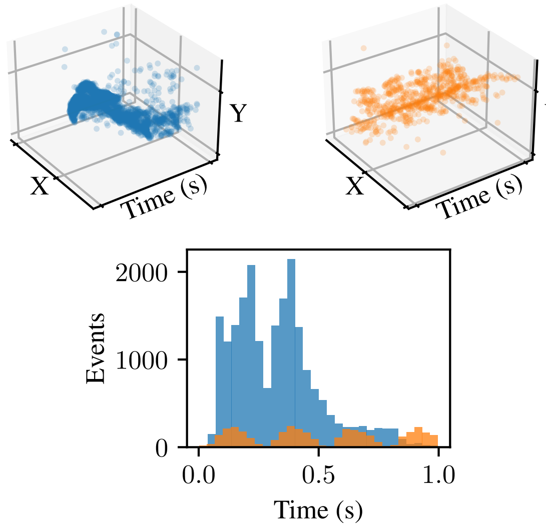

Figure 3. Events generated by a flying bee (left, blue) and a twig moving in the wind (right, orange) during a time of one second. Both objects show similar displacements, but the event count of the bee is significantly higher (bottom).

Insect classification When an object of fixed shape moves, events are primarily generated from contrast changes of the leading and trailing edges of their silhouette and texture [19], whereas most events generated by a flying insect are generated by the wing movements. This observation results in the heuristic that insects generate a significantly higher rate of events per occupied pixel than moving plants for a given movement speed (see Fig. 3). These properties allow the classification of clusters representing insects versus other moving objects like plants and debris by tracking their speed, event rate and shape over time.

To integrate this observation the exponentially weighted averages of the cluster movement vector m and of the foreground confidence f were added as additional cluster attributes and iteratively updated over time analogous to Eqs. (1) and (3). The foreground confidence increases if the activity of a cluster is high relative to its area and movement speed $\left(\frac{\bar{\mathtt{a}}_t^{\mathtt{cl}}}{A^2}|m|^{-\frac{1}{2}} > \mathtt{foreground\_threshold}\right)$ and decreases otherwise. Clusters with a foreground confidence above zero are then classified as a flying insect, which can be used to trigger the temporally synchronized RGB camera to capture images.

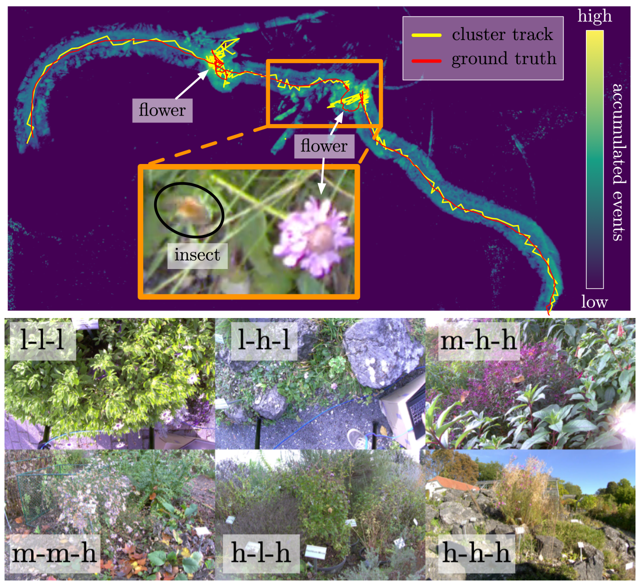

Figure 4. Accumulated event rate over 10 seconds of a bee visiting two flowers with cluster trajectory and ground truth overlay (top). First frame of every evaluated scene in order of Tab. 1 (bottom). The scene naming scheme is described in Sec. 2.2 and the caption of Tab. 1.

2.4. Identifying Suitable Parameters

As described above several parameters can be set to identify insect movement in the DVS event stream. The choice of these parameters mainly depend on the three criteria used to characterise the challenges in the different scenes (see Sec. 2.2). To evaluate both, the best possible performance given optimal parameters for each scene and a set of fixed parameters as a trade-off for all scenes we performed extensive parameter scans for these conditions. In particular we used a random-search strategy to identify scene-specific and global parameter configurations and evaluated our pipeline using these sets (cf. Sec. 3).

3. Results

Our overall goal is to demonstrate that DVS are suitable to detect insect presence in cluttered natural environments (e.g. to trigger the RGB camera). Insect detections are computed using iteratively updated temporal and spatial information in the event stream and insect locations can be identified based on cluster centres in the DVS signal. Therefore, we evaluate both the detection accuracy on a per-frame basis (i.e. correctly identified frames with or without insects using the DVS as an indicator) and the detection F1 score based on bounding boxes in discretised DVS images (see Sec. 2.2). To analyse the impact of the user-specified parameters (see Sec. 2) we evaluated our algorithm using an optimal set per scene (called scene-specific configuration) and one fixed set used for all scenes (called global configuration).

In addition, we compare the results of our algorithm to two deep learning-based approaches, namely YOLOv7 as a baseline for state-of-the-art object detectors [37] and EVPropNet as a baseline for for event-based object detectors [32]. Since EVPropNet targets the detection of rotating drone propellers it shares similarities to our goal of detection insect wings given comparable activity densities. Given our focus to evaluate the DVS data for efficient RGB camera triggering and to enable direct comparability we evaluate both approaches on event data only.

For EVPropNet the original pretrained model by Sanket *et al.* was used [32] on the full frame resolution of the accumulated events between trigger signals. Predictions were generated by locating local maxima in the heatmap-like output of the network. The confidence threshold determining whether a peak should be considered an insect prediction was chosen per scene to yield the best results. We therefore compare our work to the best possible performance of pretrained EVPropNet. The inference of one image on a reference consumer grade graphics card (RTX 2060 Super) takes approximately 31.6ms on average. All results regarding EVPropNet can be found in Tab. 2.

For YOLOv7 the official implementation [37] was used in all experiments and we chose the standard v7 model instead of the bigger variants such as YOLOv7-e6e since fast inferences are required to trigger the RGB camera in case of insect visitations. Again accumulated events between trigger signals at full spatial resolution were used as an input for training and testing. YOLOv7 was trained for 50 epochs with a batch size of 16. The best model was chosen according to the weighted combination of $0.1 \cdot$ mAP@0.5 $+ 0.9 \cdot$ mAP@0.5:0.95, as is proposed in the official implementation. In order to evaluate the performance 6-fold cross validation was used on the six scenes of our dataset resulting in 6 models optimized on five scenes with the sixth scene used for testing only. The inference time of one image for a traced YOLOv7 model is approximately 22.9ms on average on a reference consumer grade graphics cart (RTX 2060 Super). We also evaluated the impact of reduced image sizes on the computational performance. Given the already small object sizes (cf. Tab. 1) the reductions however resulted in unstable network trainings. Moreover, we tested to only reduce the resolution during inference which resulted in a large drop in detection performance. Given these results and the inference time on the reference consumer graphics card we conclude that it would be difficult to decrease the inference time much further on edge devices that would be suited for remote outdoor recording scenarios. As for EVPropNet we chose the confidence thresholds for YOLOv7 to be optimal for every scene. All results for YOLOv7 are summarised in Tab. 2.

Scene (dst-fd-bd)	Ours (Global Configuration)								
	Box F1	Box Prec	Box Rec	Frame Acc	Frame Acc + δ	Frame F1	Frame Prec	Frame Rec	Proc. Speed
l-l-l	69.62	88.20	57.50	76.18	67.59	71.10	92.31	57.81	5111
l-h-l	63.49	60.24	67.10	96.84	89.95	67.99	66.26	69.82	4860
m-h-h	50.24	90.99	34.70	67.55	85.75	75.10	98.68	60.61	4961
m-m-h	40.00	82.97	26.35	59.38	69.70	53.12	97.67	36.48	2959
h-l-h	27.83	36.13	22.63	94.54	88.89	45.30	60.27	36.29	9183
h-h-h	22.35	35.81	16.24	48.18	51.37	39.75	75.04	27.04	1048
avg	45.59	65.72	37.42	73.78	75.54	58.73	81.70	48.01	4687
Scene (dst-fd-bd)	**Best YOLOv7 Configuration**								
	Box F1	Box Prec	Box Rec	Frame Acc	Frame Acc + δ	Frame F1	Frame Prec	Frame Rec	Conf. Thresh
l-l-l	64.56	81.83	53.31	74.87	73.46	68.59	93.55	54.15	7.3
l-h-l	58.51	59.68	57.39	96.41	79.22	61.13	63.72	58.75	24.6
m-h-h	46.03	69.89	34.31	69.16	87.98	76.76	98.00	63.08	4.4
m-m-h	37.12	58.23	27.24	57.73	68.45	54.82	84.15	40.65	15.9
h-l-h	46.07	39.77	54.74	95.00	57.73	57.95	60.77	55.38	5.4
h-h-h	24.32	28.58	21.16	45.23	58.80	44.29	62.06	34.43	19.8
avg	46.10	56.33	41.36	73.06	70.94	60.59	77.04	51.07	12.9
Scene (dst-fd-bd)	**Best EVPropNet Configuration**								
	Box F1	Box Prec	Box Rec	Frame Acc	Frame Acc + δ	Frame F1	Frame Prec	Frame Rec	Conf. Thresh
l-l-l	50.72	67.56	40.60	68.44	72.78	58.70	87.11	44.27	3.92
l-h-l	6.17	3.65	19.98	80.19	31.39	11.92	7.58	27.89	68.63
m-h-h	7.87	5.68	12.77	58.68	85.64	67.82	91.35	53.93	3.92
m-m-h	7.19	4.86	13.80	43.43	67.86	43.39	58.85	34.36	3.92
h-l-h	1.02	14.29	0.53	93.74	87.90	1.06	33.33	0.54	3.92
h-h-h	1.70	0.99	6.07	32.58	56.42	36.76	45.18	30.99	3.92
avg	12.45	16.17	15.63	62.84	67.00	36.61	53.90	32.00	14.71

Table 2. Results overview. All values (see Sec. 3) are given in percent. $\delta(= 100)$: Number of additional frames automatically captured after each assumed detection. The scene naming scheme is described in Sec. 2.2 and the caption of Tab. 1.

Scene (dst-fd-bd)	Ours (Scene Specific Configuration)			
	Box F1	Frame Acc	Frame Acc + δ	Frame F1
l-l-l	70.27	76.37	68.88	71.01
l-h-l	74.49	98.07	95.70	76.00
m-h-h	56.57	80.40	84.85	87.20
m-m-h	40.17	60.32	65.93	56.34
h-l-h	32.47	93.98	84.24	44.44
h-h-h	27.43	52.62	58.76	48.22
avg	50.23	76.96	76.39	63.87

Table 3. Results Ours with configurations optimized for each individual scene. All values (see Sec. 3) are given in percent. $\delta(= 100)$: Number of additional frames automatically captured after each assumed detection. The scene naming scheme is described in Sec. 2.2 and the caption of Tab. 1.

Frame-based Evaluation We define a frame as true positive if it contains an insect and the corresponding event stream comprises an insect cluster. Likewise, true negative frames neither contain an insect nor a cluster, false negative frames include an insect but no cluster, and false positive frames do not include insects, but an insect cluster has been identified in the DVS stream. This definition is used to calculate the accuracy, precision, recall and F1 score on a per-frame level based on the confident user annotations only.

As can be seen in Tab. 2 the frame accuracy varies between the different scenes, ranging from 96.84% to 48.18% with an overall mean accuracy of 73.78% between scenes. Over the total number of frames between scenes, 81.02% of all RGB images (recorded with $\sim$ 100fps) are correctly identified as either containing an insect or are correctly classified as empty if triggering would be done on a per-frame basis. This is also reflected in the F1 score and the corresponding precision and recall values. Assuming a triggering mechanism as used for conventional camera traps, in which a detected animal initiates the collection of a continuous image sequence of δ frames (in our case 1 second, resulting in $\delta = 100$) the average frame accuracy increases to 75%.

Given the different scene complexities, the distance to the area of interest (c.f. Tab. 1) has the strongest impact on the frame-based accuracy values. Using parameters optimised for particular scenes (Tab. 3), minor improvements can be recognized. However, given that this distance is inversely proportional to the number of pixels covered by the appearance of the insects, smaller distances are recommended as expected.

As can be seen in Tab. 2 our algorithm outperforms YOLOv7 and EVPropNet in almost all global and scene-specific configurations for the frame-based metrics. Only for the scene "h-l-h" YOLOv7 outperforms our approach (for a discussion see bounding box-based evaluation below).

Bounding Box-based Evaluation To extract the localisation score in the event stream, cluster centres $\mu^{\mathtt{cl}}$ are matched to user-specified bounding boxes by distance. Centres inside a minimum enclosing circle of the bounding box are considered a true positive, cluster centres with no corresponding bounding box specify false positives and bounding boxes without a corresponding cluster are considered false negatives. These estimates are used to quantify the precision, recall and F1 score using the confident user annotations only.

An exemplary accumulated event image with cluster centre and ground truth trajectory is given in Fig. 4. As can be seen in Tab. 2 and similar to the frame-based evaluation, the F1 score varies between 69.62% and 22.35%. The distance to the area of interest has the strongest impact but the dynamics within this area and the background are also impacting our object detection performance. Tuning the parameters for individual scenes again improves the F1 score for boxes; however, given the highly complex and challenging scenarios, an average precision of 65.72% and recall of 37.42% for our global configuration demonstrate that the DVS-based object detection is generalising well enough for initial insect localisation.

As can be seen in Tab. 2 our algorithm outperforms YOLOv7 and EVPropNet on the bounding box-based metrics. Similar to the frame-based evaluation, YOLOv7 achieves the best scores for the "h-l-h" scene. A possible explanation could be the particularly small appearance of the insects in this scene (see Tab. 1). As a consequence, the body and wings of the insects do not generate sufficient event counts in the DVS stream. For YOLOv7 this is not necessarily important, since it uses images composed of accumulated events, which do not depend on the actual event activity per pixel but is mostly influenced by the shape of the object of interest. Instead the performance of YOLOv7 is more dependent on a combination of the distance to the area of interest, the movement of objects with insect like shapes in the accumulated event stream, and the overall movement around insects.

Processing Speed Enabling DVS-based triggering requires event processing faster than the frame rate of the RGB camera (here: 100fps). In this context, we define processing speed as the ratio between recording duration and the amount of time necessary for a given algorithm to fully analyse the recording. As can be seen in column 'Proc. Speed' of Tab. 2 events are processed on average with 4800% of the necessary real-time performance as a single threaded workload on a AMD Ryzen 3700X CPU. Since the number of events increases with the dynamics of the scene and the background (cf. Tab. 1), the processing speed decreases to 1048%, which is still sufficient to trigger a high frame-rate RGB camera. The processing speed of the deep learning-based approaches does not depend on the event rate, but is still much slower than our algorithm. In direct comparison, YOLOv7 outperforms EVPropNet but still requires 22.9ms per frame on an RTX 2060 Super, which equates to 43.67% real-time performance. Moreover, the higher power requirements of deep learning-based approaches are also disadvantageous compared to the proposed solution.

4. Conclusion

In this paper, we describe our progress towards building a DVS-based insect camera trap. In particular, we introduce this type of sensing in combination with a real-time small object detection algorithm to identify insect presence beyond the capabilities of conventional imaging. Our evaluations demonstrate that the intrinsic benefits of event cameras are indeed favourable to disambiguate background from insect motion, while asynchronous processing enables insect localisation at very high event rates (beyond the speed of consumer RGB cameras). A comparison against two state-of-the-art deep learning models revealed superior detection performance and lower processing times.

The high dynamic range and low energy consumption of DVS are also favourable characteristics for outdoor applications, which we are going to elaborate on in more detail in the future. We will also investigate the combination of RGB and DVS cues for more targeted insect localisation purposes and extend our evaluations towards nocturnal insects. Moreover, we will investigate if combined imaging cues will be beneficial for species classification tasks. We conclude that this type of neuromorphic sensing has the potential to initiate the RGB image-capturing process and could therefore be an important technology for building a general-purpose insect camera trap.

Acknowledgments

The training of deep learning algorithms was performed on the computer cluster PALMA II of the University of Münster. All authors would like to thank the Botanical Garden of the University of Münster, the Center for Nonlinear Science, Jacqueline Kockwelp and the Human Frontier Science Program [RGP0057/2021].

References

[1] Marcelo A Aizen, Lucas A Garibaldi, Saul A Cunningham, and Alexandra M Klein. How much does agriculture depend on pollinators? lessons from long-term trends in crop production. *Annals of botany*, 103(9):1579–1588, 2009. 1

[2] Ignacio Alzugaray and Margarita Chli. Asynchronous corner detection and tracking for event cameras in real time. *IEEE Robotics and Automation Letters*, 3(4):3177–3184, 2018. 2, 4

[3] Don Chathurika Kshanthi Amarathunga, John Grundy, Hazel Parry, and Alan Dorin. Methods of Insect Image Capture and Classification: A Systematic Literature Review. *Smart Agricultural Technology*, 1:100023, 2021. 1

[4] Yancheng Bai, Yongqiang Zhang, Mingli Ding, and Bernard Ghanem. Finding tiny faces in the wild with generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 21–30, 2018. 2

[5] Francisco Barranco, Cornelia Fermuller, and Eduardo Ros. Real-Time Clustering and Multi-Target Tracking Using Event-Based Sensors. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 00:5764–5769, 2018. 2

[6] Simone Belluco, Michela Bertola, Fabrizio Montarsi, Guido Di Martino, Anna Granato, Roberto Stella, Marianna Martinello, Fulvio Bordin, and Franco Mutinelli. Insects and public health: An overview. *Insects*, 14(3):240, 2023. 1

[7] Kim Bjerge, Hjalte M. R. Mann, and Toke Thomas Høye. Real-time insect tracking and monitoring with computer vision and deep learning. *Remote Sensing in Ecology and Conservation*, 8(3):315–327, 2022. 1

[8] Tobias Bolten, Regina Pohle-Frohlich, and Klaus D Tonnies. Dvs-outlab: A neuromorphic event-based long time monitoring dataset for real-world outdoor scenarios. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1348–1357, 2021. 2

[9] Changrui Chen, Yu Zhang, Qingxuan Lv, Shuo Wei, Xiaorui Wang, Xin Sun, and Junyu Dong. Rrnet: A hybrid detector for object detection in drone-captured images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 2

[10] Raphael K Didham, Yves Basset, C Matilda Collins, Simon R Leather, Nick A Littlewood, Myles HM Menz, Jörg Müller, Laurence Packer, Manu E Saunders, Karsten Schönrogge, et al. Interpreting insect declines: seven challenges and a way forward. *Insect Conservation and Diversity*, 13(2):103–114, 2020. 1

[11] Vincent Droissart, Laura Azandi, Eric Rostand Onguene, Marie Savignac, Thomas B. Smith, and Vincent Deblauwe. PICT: A low-cost, modular, open-source camera trap system to study plant–insect interactions. *Methods in Ecology and Evolution*, 12(8):1389–1396, 2021. 1

[12] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J Davison, Jörg Conradt, Kostas Daniilidis, et al. Event-based vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(1):154–180, 2020. 2

[13] Neil A Gilbert, John DJ Clare, Jennifer L Stenglein, and Benjamin Zuckerberg. Abundance estimation of unmarked animals based on camera-trap data. *Conservation Biology*, 35(1):88–100, 2021. 1

[14] Yuqi Gong, Xuehui Yu, Yao Ding, Xiaoke Peng, Jian Zhao, and Zhenjun Han. Effective fusion factor in fpn for tiny object detection. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1160–1168, 2021. 2

[15] Lars Haalck, Michael Mangan, Barbara Webb, and Benjamin Risse. Towards image-based animal tracking in natural environments using a freely moving camera. *Journal of Neuroscience Methods*, 330:108455, 2020. 1

[16] Michael T Hobbs and Cheryl S Brehme. An improved camera trap for amphibians, reptiles, small mammals, and large invertebrates. *PloS one*, 12(10):e0185026, 2017. 1

[17] Peiyun Hu and Deva Ramanan. Finding tiny faces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 951–959, 2017. 2

[18] Jianhao Jiao, Huaiyang Huang, Liang Li, Zhijian He, Yilong Zhu, and Ming Liu. Comparing representations in tracking for event camera-based slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1369–1376, 2021. 2

[19] Xavier Lagorce, Cédric Meyer, Sio-Hoi Ieng, David Filliat, and Ryad Benosman. Asynchronous Event-Based Multikernel Algorithm for High-Speed Visual Features Tracking. *IEEE Transactions on Neural Networks and Learning Systems*, 26(8):1710–1720, 2015. 2, 5

[20] Hei Law and Jia Deng. CornerNet: Detecting objects as paired keypoints. *ECCV*, 2016. 2

[21] Jianan Li, Xiaodan Liang, Yunchao Wei, Tingfa Xu, Jiashi Feng, and Shuicheng Yan. Perceptual generative adversarial networks for small object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1222–1230, 2017. 2

[22] Gen Liu, Jin Han, and Wenzhong Rong. Feedback-driven loss function for small object detection. *Image and Vision Computing*, 111:104197, 2021. 2

[23] Robert W Matthews and Janice R Matthews. The malaise trap: its utility and potential for sampling insect populations. *The Great Lakes Entomologist*, 4(4):4, 2017. 1

[24] Anindya Mondal and Mayukhmali Das. Moving Object Detection for Event-based Vision using k-means Clustering. *2021 IEEE 8th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, 00:1–6, 2021. 2

[25] Elias Mueggler, Chiara Bartolozzi, and Davide Scaramuzza. Fast Event-based Corner Detection. *Procedings of the British Machine Vision Conference 2017*, 2017. 2

[26] Manasi Muglikar, Mathias Gehrig, Daniel Gehrig, and Davide Scaramuzza. How to Calibrate Your Event Camera. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 00:1403–1409, 2021. 3

[27] Qaim Naqvi, Patrick J. Wolff, Brenda Molano-Flores, and Jinelle H. Sperry. Camera traps are an effective tool for

monitoring insect–plant interactions. *Ecology and Evolution*, 12(6):e8962, 2022. 1

[28] Simon G Potts, Vera Imperatriz-Fonseca, Hien T Ngo, Marcelo A Aizen, Jacobus C Biesmeijer, Thomas D Breeze, Lynn V Dicks, Lucas A Garibaldi, Rosemary Hill, Josef Settele, et al. Safeguarding pollinators and their values to human well-being. *Nature*, 540(7632):220–229, 2016. 1

[29] Malika Nisal Ratnayake, Adrian G. Dyer, and Alan Dorin. Towards Computer Vision and Deep Learning Facilitated Pollination Monitoring for Agriculture. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 00:2915–2924, 2021. 1

[30] Malika Nisal Ratnayake, Adrian G. Dyer, and Alan Dorin. Tracking individual honeybees among wildflower clusters with computer vision-facilitated pollinator monitoring. *PLOS ONE*, 16(2):e0239504, 2021. 1

[31] J.P. Rodríguez-Gomez, A. Gómez Eguíluz, J.R. Martínez-de Dios, and A. Ollero. Asynchronous event-based clustering and tracking for intrusion monitoring in UAS. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 00:8518–8524, 2020. 2

[32] Nitin J Sanket, Chahat Deep Singh, Chethan Parameshwara, Cornelia Fermüller, Guido de Croon, and Yiannis Aloimonos. EVPropNet: Detecting Drones By Finding Propellers For Mid-Air Landing And Following. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. 2, 3, 6

[33] Gianluca Scarpellini, Pietro Morerio, and Alessio Del Bue. Lifting monocular events to 3d human poses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1358–1368, 2021. 2

[34] Sebastian Seibold, Martin M. Gossner, Nadja K. Simons, Nico Blüthgen, Jörg Müller, Didem Ambarlı, Christian Ammer, Jürgen Bauhus, Markus Fischer, Jan C. Habel, Karl Eduard Linsenmair, Thomas Nauss, Caterina Penone, Daniel Prati, Peter Schall, Ernst-Detlef Schulze, Juliane Vogt, Stephan Wöllauer, and Wolfgang W. Weisser. Arthropod decline in grasslands and forests is associated with landscape-level drivers. *Nature*, 574(7780):671–674, 2019. 1

[35] Schraml Stephan and Belbachir Ahmed Nabil. A Spatio-temporal Clustering Method Using Real-time Motion Analysis on Event-based 3D Vision. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pages 57–63, 2011. 2, 4

[36] David L Wagner, Eliza M Grames, Matthew L Forister, May R Berenbaum, and David Stopak. Insect decline in the anthropocene: Death by a thousand cuts. *Proceedings of the National Academy of Sciences*, 118(2):e2023989118, 2021. 1

[37] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv*, 2022. 2, 3, 6

[38] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as Points. *arXiv*, 2019. 2

[39] Alex Zihao Zhu, Nikolay Atanasov, and Kostas Daniilidis. Event-Based Feature Tracking with Probabilistic Data Association. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4465–4470, 2017. 2