# A motion-based compression and tracking system for video camera trap-based insect behaviour studies

Malika Nisal Ratnayake[1*], Lex Gallon[1,2], Adel N Toosi[3,4], Alan Dorin[1]

[1*]Dept. of Data Science and AI, Faculty of Information Technology, Monash University, Clayton, 3800, Victoria, Australia.
[2]Monash e-Research Centre, Monash University, Clayton, 3800, Victoria, Australia.
[3]School of Computing and Information Systems, Faculty of Engineering and Information Technology, The University of Melbourne, Parkville, 3052, Victoria, Australia.
[4]Dept. of Software Systems and Cybersecurity, Faculty of Information Technology, Monash University, Clayton, 3800, Victoria, Australia.

*Corresponding author(s). E-mail(s): malika.ratnayake@monash.edu;
Contributing authors: lex.gallon@monash.edu; adel.toosi@unimelb.edu.au;
alan.dorin@monash.edu;

**Abstract**

Field-captured video enables detailed study of animal locomotion, decision-making, and environmental interactions such as predator–prey dynamics and habitat use. While low-cost hardware makes data capture accessible, the storage, processing, and transmission demands of high-resolution video remain a major hurdle for field-deployed edge computing devices. Motion tracking in natural environments presents unique challenges that require tailored video compression strategies not well addressed in other domains. We present a novel end-to-end system comprising a motion analysis-based video compression algorithm specifically designed for camera traps, and a custom video processing methodology for automated analysis of compressed footage to extract behavioural data. We evaluate it through a case study on insect–pollinator motion tracking using three popular edge computing platforms. The compression algorithm operates alongside standard codecs, identifying and storing only image regions containing motion relevant to monitoring tasks, reducing data size by an average of 87% across diverse datasets. When combined with the H.265/HEVC codec, our approach achieved an additional 47.1% improvement in compression compared to stand-alone H.265. The accompanying video processing algorithm builds upon existing Polytrack software, incorporating new preprocessing and trajectory reconstruction techniques for automated processing of compressed footage with a 97.5% detection rate. Our experiments demonstrate that the system retains critical behavioural information, as verified through both automated and manual analyses. The method presented in this paper enhances the applicability of low-powered computer vision edge devices to remote, *in situ* animal motion monitoring, and improves the efficiency of playback during behavioural analyses.

**Keywords:** camera traps, edge computing, wildlife monitoring, video compression, motion analysis, insect monitoring, insect tracking

# 1 Introduction

Camera traps are indispensable to monitor how increasing environmental change impacts animal behaviour (Caravaggi et al., 2017), habitat use (Dharmarathne, Jayasekara, Mahaulpatha, & de Silva, 2022; Head, Robbins, Mundry, Makaga, & Boesch, 2012; Lovell, Li, Turner, & Carbone, 2022), species abundance and distributions (Feng et al., 2021; Reece, Radloff, Leslie, Amin, & Tambling, 2021). Currently, their use to collect high resolution video in remote areas is limited by transmission bandwidth constraints when data is transferred over wireless cellular or satellite networks (Wong & Kachel, 2024). Therefore, enhancements in data compression, processing, and transmission, are required to capitalise on the availability of cheap data capture devices. In this article, we explore this general problem by highlighting properties specific to data-rich video camera traps for wildlife monitoring. These needs suggest avenues to improve monitoring device autonomy that are arguably more important for this domain than in the domains dominating algorithm development, such as Closed-circuit television (CCTV) for security and surveillance.

Animal video monitoring often involves long periods of inactivity, punctuated by bursts of activity that can be difficult to predict or capture. This requires us to consider which periods of video data are essential to record, maintain, and which might be discarded or compressed. The need to survey a wide field of view to capture animal movement conflicts with requirements for detailed, focused data to identify and track individuals, especially small, fast animals like insects (Ratnayake, Amarathunga, Zaman, Dyer, & Dorin, 2023). This requires us to consider which sections of an individual video frame must be maintained, and which might be compressed or discarded. Environmental variations, such as the arrival or loss of flowers, water and shade, influence animal behaviours. This requires an accurate record of the environment *and* animals to analyse their interactions. Together, these factors present an opportunity to develop specific techniques for data capture, compression, management and analysis.

Camera traps can collect still images or video. Still image traps capture single or short-burst sequences at predetermined intervals or when triggered (Collett & Fisher, 2017). Video traps may record continuous, sometimes lengthy, image sequences at sampling rates above 30 fps at preset intervals, or when triggered. Video data allows analysis of dynamic interactions (Green, Stephens, Whittingham, & Hill, 2023; Janisch et al., 2021; Rampim, Sartorello, Fragoso, Haberfeld, & Devlin, 2020), including animal pollination (Krauss, Roberts, Phillips, & Edwards, 2018; Melidonis & Peter, 2015; Ratnayake et al., 2023), that are not fully captured in still images. However, video is costly to store, process, and transmit from remote devices with limited access to power, storage, and transmission bandwidth. Some systems, therefore, run for limited periods, avoid onboard processing, and require manual data transfer (Droissart et al., 2021). This can reduce their value. Ideally, traps would be capable of remote autonomous operation with infrequent service visits, and all data would be streamed conveniently to the cloud. It is this need for increased camera trap autonomy that drives us to reduce storage and transmission bandwidth requirements by compressing video data. We apply a computationally low-cost algorithm modified specifically for the situation and demonstrate its benefits on insect pollinator monitoring as a case study.

Insect pollinator monitoring is a valuable case study as pollination underpins natural ecosystems (Food & Agriculture Organization, 2019) and sustains global food production (Gazzea, Batáry, & Marini, 2023). Insects' natural foraging environments are subject to frequent change due to wind and sunlight. Insect species vary widely in size, morphology, microhabitat preferences, and motion characteristics. All of these factors contribute to the difficulty of monitoring insects, making video capture at high spatio-temporal resolution invaluable, but as noted, costly (Nykänen, Pöysä, Matala, & Kunnasranta, 2023; Ratnayake et al., 2023; Sheard et al., 2024; Van Klink et al., 2024).

In this paper, we introduce software requiring low computational resources that substantially compresses videos without compromising insect monitoring data. We process videos frame-by-frame and pixel-by-pixel to identify image regions depicting motion, and store only information relevant for animal identification and for constructing motion paths and habitat (flower) interactions. The approach can be seamlessly integrated

with common video compression (lossless and lossy) and recording methods (continuous, time-lapse, motion-triggered) to improve performance. In addition, we extend the existing open-source outdoor insect tracking software Polytrack (Ratnayake et al., 2023) with the method to extract and reconstruct motion trajectories from compressed videos, enabling automated behavioural analysis. By combining edge video compression with Polytrack video analysis, we establish a camera trap-based monitoring pipeline for automated video compression and analysis. We tested our algorithm on six datasets from diverse application environments. Alongside this article, we publish the compression algorithm (*EcoMotionZip*), documentation, and extended Polytrack software.

# 2 Related Work

Storage optimisation in resource-constrained video camera traps can be achieved in two ways. Video compression optimisation reduces file sizes by removing unimportant information while preserving regions of interest in high detail. This approach retains the video's full temporal information. Event-driven data collection uses time-lapse or motion-triggered recording to minimise the storage of irrelevant or non-informative data. However, this method captures only spatio-temporal snapshots of the observed scene. In practice, a combination of these two strategies is commonly employed in video camera trapping to minimise storage requirements while retaining critical data.

## 2.1 Video compression optimisation

Lossless video compression preserves all data, ensuring high-quality output but this may result in large file sizes. Lossy compression reduces file size by discarding what are considered to be unimportant image details, attempting to strike a balance between image quality and storage efficiency. As a result, lossy compression is commonly used in video camera traps. Such compression is typically achieved through the use of video codecs, which encode raw video streams into more compact representations. Early camera trap systems employed Motion JPEG (MJPEG) (Pennebaker & Mitchell, 1992), a legacy codec that applies JPEG compression independently to each frame (Fairhurst, Nazir, & Verdicchio, 2013). More advanced codecs have since been adopted. Advanced Video Coding (AVC / H.264) (Wiegand, Sullivan, Bjontegaard, & Luthra, 2003) has become widely used in edge computing–based camera traps, primarily due to its broad hardware acceleration support and compatibility across devices. More recently, High Efficiency Video Coding (HEVC / H.265) (Sullivan, Ohm, Han, & Wiegand, 2012) has been introduced as the successor to H.264, offering significantly improved compression while maintaining comparable quality. However, the computational complexity of HEVC and the limited availability of hardware acceleration on low-power devices have restricted its adoption in resource-constrained edge environments.

Recent research has proposed methods to improve the processing speed and compression of lossy and lossless algorithms. For example, mean predictive block matching is designed to enhance lossy and lossless compression in human video surveillance (Ahmed et al., 2020). However, while effective for human surveillance in urban environments, this is unsuitable for insect monitoring because insects make erratic movements that are hard to predict, reducing the algorithm's performance.

Region of Interest (ROI)-based encoding, such as ROI-based neural video coding, is an effective technique for video compression (Dou, Cao, & Zhang, 2024; Le et al., 2022; Liu, Li, Lin, Li, & Wu, 2020; Perugachi-Diaz et al., 2022; Ungureanu, Negirla, & Korodi, 2024). This method identifies any ROI within a video frame and encodes it at a high bitrate, while less important areas are encoded at a lower bitrate to reduce file size. Neural network-based approaches are commonly used for ROI encoding to adapt to the variability of recording environments. However, this technique makes high computational demands and is therefore not always suitable for remote camera traps (Liu et al., 2020). It may also fail to identify animal species that it has not been trained to recognise, and may discard potentially valuable background information.

## 2.2 Event-driven data collection

Continuous video recording requires significant storage even during periods of no animal activity. Time-lapse and scheduled recordings conserve storage by capturing videos at intervals, but risk missing relevant activity (Wittmann, Gamal Ibrahim, Straw, Klein, & Staab, 2024). Reducing temporal resolution reduces file sizes effectively for slow-moving animals but fails to capture the rapid and erratic movements of fast-moving species like insects (Schindler & Steinhage, 2021). Other video compression methods produce a "summary" of multiple motion paths or animal appearances by simultaneously displaying events that in reality spanned a long period of time (Peleg & Pritch, 2012). While effective for reducing video review time, this approach is unsuitable for animal behaviour studies as it does not preserve environmental changes or the temporal relationships between potentially interacting organisms and habitat features.

Hardware or software motion-triggers aim to optimise resource usage by activating recording only when animal activity is detected. Commonly used hardware triggers, such as Passive Infra-Red (PIR) sensors, are ineffective for detecting small insects (Naqvi, Wolff, Molano-Flores, & Sperry, 2022; Ortmann & Johnson, 2021). Software-based triggers employing deep learning (Sittinger, Uhler, Pink, & Herz, 2024) can significantly reduce storage requirements by recording only when animals are detected. However, these models typically require substantial computational resources, specialised hardware, and extensive training datasets. The reliance on training data can lead to false negatives where the presence of new species is overlooked. This issue is particularly pronounced in insect monitoring, as insects are small and often possess subtle morphological features that make them difficult to detect in natural environments. Alternatively, foreground-background segmentation-based triggers (van der Voort, Gilmore, Gorrell, & Janes, 2022) are computationally efficient but susceptible to false positives caused by environmental wind or lighting changes, leading to the recording of video segments without animal activity (Swinnen, Reijniers, Breno, & Leirs, 2014). Despite this limitation, their ability to efficiently record extensive data without significant loss make them suitable for edge devices and a good choice for animal video monitoring.

Video compression techniques for camera traps in animal monitoring must reduce file size while preserving spatiotemporal information about animal morphology, behaviour, interactions, and the environment. These techniques should enable both human and automated analysis, be computationally efficient for deployment on edge devices, and retain detail for reconstructing animal motion and ecological interactions. Existing algorithms fail to meet these specific requirements. To address this, we propose *EcoMotionZip*. This method is specifically designed for ecological video camera traps. It leverages computationally inexpensive motion analysis principles to achieve efficient compression without compromising critical spatiotemporal information.

# 3 Materials and Methods

In this section, we describe the system architecture (Fig. 1). The proposed complete system consists of two components; (i) edge video compression (Sec. 3.1) and (ii) offline video processing and behavioural analysis (Sec. 3.2).

## 3.1 Edge video compression algorithm

In this section we describe the algorithm's multi-threaded architecture designed to compress videos and improve data throughput in edge devices.

### 3.1.1 Reader component

The Reader captures video frames from a camera stream or pre-recorded video file and adds them to a queue for processing. Once all frames from the input stream have been passed to the Motion Analysis component for processing, the Reader terminates.

### 3.1.2 Motion Analysis component

The Motion Analysis component processes frames captured by the Reader thread to identify regions with motion. It is designed to extract only information from video frames critical for animal behaviour analysis. It discards the remaining information to save storage space. Maintained data includes information for algorithms (and

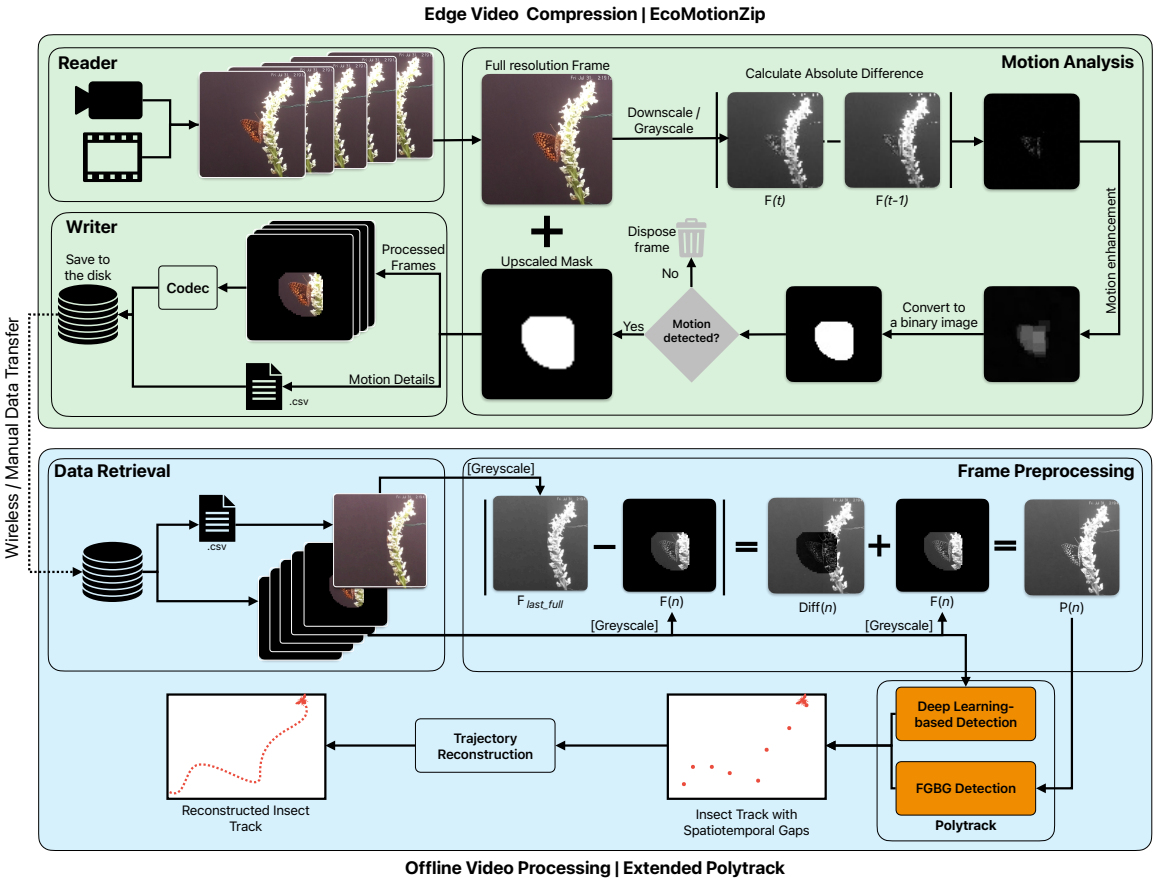



**Fig. 1**: **System Overview comprising two components. (i) edge video compression *(EcoMotionZip)* and (ii) offline video processing *(Extended Polytrack).*** EcoMotionZip includes three sub-components: (1) Reader, (2) Motion Analysis, and (3) Writer, each implemented as a separate thread. Videos compressed at the edge are transferred wirelessly or manually for offline analysis. The offline pipeline uses an extended version of Polytrack (Ratnayake et al., 2023), incorporating frame preprocessing and trajectory reconstruction to enable automated insect tracking and behaviour analysis on EcoMotionZip-compressed videos. "Deep learning–based Detection" and "FGBG Detection" denote the HyDaT detection models (Ratnayake et al., 2021b) used within Polytrack. Dots along tracks indicate recorded and estimated insect positions. Example footage is from van der Voort et al. (2022).

human viewers) to gauge (1) animal type / species, (2) movement paths / gaits, (3) observation time, and (4) a snapshot of the environment within the camera view.

Motion analysis begins by down-scaling each captured frame and converting it to greyscale to reduce computational load and improve processing efficiency (Bjerge, Frigaard, & Karstoft, 2023; Ratnayake, Dyer, & Dorin, 2021a). Subsequently, inter-frame changes are detected by calculating the absolute intensity difference between pixels in adjacent frames. Pixel regions displaying an absolute intensity difference greater than a user-set threshold are preserved and expanded to include a surrounding buffer region. Users can adjust the sensitivity of the motion capture by modifying the threshold value to suit the monitored environment and target species. The buffer region allows the algorithm to maintain detailed information about

the animal and its immediate surroundings, facilitating subsequent behavioural analysis. The size of the buffer region can be customised by the user. Subsequently, the frame containing motion regions is converted into a binary image, where pixels of non-zero intensity are retained while the rest are zeroed. Frames with no regions of motion are discarded completely. Frames with regions of detected motion undergo upscaling of the binary image to the original frame size and a bitwise product is generated between the upscaled binary image and the original frame. The resultant frame is then passed to the Writer component for storage. The Motion Analysis component terminates once all frames from the Reader have been processed. In addition, the algorithm captures full frames at the start and at user-specified intervals during motion sequences to provide an overview of the scene and document any gradual changes in the environment that occur over the recording period. Alongside processed motion frames sent to the Writer, the Motion Analysis component transmits the frame numbers of the input and output videos, and whether or not a full image frame has been saved. This information is stored in a CSV file for later reconstruction of animal motion and behaviour.

#### 3.1.3 Writer component

The Writer component receives processed frames from the Motion Analysis component and reassembles them into a video file. It extracts the codec, frame rate, and resolution from the input video and applies the same properties when generating the output. Users may opt for state-of-the-art codecs (e.g. H.264, H.265 etc.) within the Writer component to achieve further improved compression or enhanced compatibility with specific processing pipelines. Additionally, the Writer stores a CSV file alongside the video file containing the supplementary data sent by the Motion Analysis component.

#### 3.1.4 Software implementation

We have implemented the proposed algorithm as *EcoMotionZip* available on GitHub via the link provided under the Code Availability section. EcoMotionZip was developed using Python 3.11.2, Computer Vision Library (OpenCV) 4.6.0, Numpy 1.24.2, and FFMPEG 5.1.4. In addition to processing and compressing pre-recorded videos, the software supports real-time video recording and processing with the PiCamera2 library. Detail on software dependencies, installation and user guides appear on GitHub.

### 3.2 Offline video processing and behavioural analysis

The ability to automatically process camera trap data for behavioural information is crucial for analysing data collected over long periods. While current software options are limited, Polytrack software (Ratnayake et al., 2023) has shown promise in tracking multiple insect species simultaneously and monitoring their flower visiting behaviour outdoors. Polytrack features a hybrid detection and tracking (HyDaT) algorithm that combines a deep learning-based detection model with a foreground-background segmentation-based detection model for insect detection (Ratnayake et al., 2021b). This hybrid approach allows Polytrack to accurately track insects in outdoor environments with high precision and recall, even when using a deep learning model trained on a limited dataset. However, Polytrack's current architecture prevents it from processing compressed videos, such as those from EcoMotionZip. We therefore incorporated new frame pre-processing and track reconstruction steps into Polytrack to handle compressed videos more effectively. The frame pre-processing step enables the extraction of insect trajectories by utilising information from compressed videos and supplementary details provided in separate CSV files. The track reconstruction step addresses spatiotemporal gaps in insect trajectories caused by dropped frames and removed pixel areas (resulting from stationary insects), generating a complete trajectory of insect movement.

#### 3.2.1 Frame pre-processing

Polytrack employs a hybrid detection and tracking model (Ratnayake et al., 2021b) that combines a deep learning-based object detector with a foreground/background (FGBG) segmentation model to track insects and flowers. However, motion regions captured in compressed videos from EcoMotionZip often include a buffer region showing the animal's immediate surroundings. This

can reduce the accuracy of insect bounding box calculations made by the FGBG segmentation detector in Polytrack. The objective of the frame pre-processing step is to eliminate the buffer region, allowing Polytrack's FGBG segmentation model to calculate a bounding box that tightly fits around the insect's body, thereby improving detection precision. Pre-processing starts by retrieving information from the CSV file about the positions of full frames in the video sequence, where snapshots of the entire scene are captured. Subsequently, frames are sequentially extracted from the video, and the full frames are stored as reference frames after their conversion to grayscale. Frames containing motion data are then processed with the stored reference frames, as outlined in Equation 1 to obtain the respective reconstructed frames with both motion and environmental pixels.

$$P(n) = \begin{cases} F(n), & n \in S_{\text{full}}, \\ F(n) + |F(n) - F_{\text{last\_full}}|, & n \notin S_{\text{full}}. \end{cases} \tag{1}$$

where $P(n)$ is the preprocessed frame, $F(n)$ denotes the current frame at time step $n$, $F_{\text{last\_full}}$ represents the most recent "full" frame, and $S_{\text{full}}$ is the set of frame indices corresponding to full frames.

The subsequent equations describe the components of Equation 1 in detail. For frames where $n \in S_{\text{full}}$, the current frame $F(n)$ is stored as the most recent full frame, $F_{\text{last\_full}}$, and directly assigned to $P(n)$:

$$F_{\text{last\_full}} \leftarrow F(n) \tag{2}$$

For frames where $n \notin S_{\text{full}}$, the absolute difference between the current frame $F(n)$ and the most recent full frame $F_{\text{last\_full}}$ is computed as:

$$\text{Diff}(n) = |F(n) - F_{\text{last\_full}}| \tag{3}$$

This difference frame is then added to the current frame to produce the preprocessed frame $P(n)$:

$$P(n) = F(n) + \text{Diff}(n). \tag{4}$$

The reconstructed frames, along with the original compressed video frames, are subsequently transferred to Polytrack for insect tracking and behaviour analysis.

### 3.2.2 Trajectory reconstruction

Insect trajectories generated from compressed videos often exhibit gaps in spatio-temporal data due to the unavailability of continuous frame sequences. These gaps can occur when insects remain stationary for an extended period, causing EcoMotionZip to interpret them as part of the background. When these stationary insects resume movement, the trajectory extraction process restarts, leaving gaps in the recorded data. Filling these gaps is essential for accurately analysing insect behaviour and ensuring the completeness of the trajectory.

To address this, a trajectory reconstruction post-processing step is applied that interpolates missing spatial ($x$, $y$) and temporal ($f$) data points.

To reconstruct missing data points in insect trajectories, the following method is used.

$$D = \{(f_i, x_i, y_i)\} \tag{5}$$

where, $D$ denotes the insect trajectories extracted with missing spatio-temporal data points, $f_i$ is the frame number in the original video, and $x_i, y_i$: spatial coordinates corresponding to the recorded frame $f_i$.

The reconstruction process consists of the following steps:

**Sorting and frame range definition**

The insect track $D$, extracted with missing spatio-temporal data, is initially sorted by frame number $f_i$. The range of frames is defined as:

$$F = \{f : f \in [f_{\text{start}}, f_{\text{end}}]\} \tag{6}$$

where, $F$ is the set of all frame numbers within the defined range $[f_{\text{start}}, f_{\text{end}}]$, $f_{\text{start}}$ is the smallest frame number in the input data $D$, and $f_{\text{end}}$ is the largest frame number in the input data $D$.

For each frame $f \in F$, the processed and interpolated data $P(f)$ is defined as:

$$P(f) = \begin{cases} (f, x, y), & \text{if } f \in \{f_i\}, \\ (f, \text{None}, \text{None}), & \text{if } f \notin \{f_i\}. \end{cases} \tag{7}$$

**Interpolation of missing coordinates**

Let $\mathcal{V} = \{f_i\}$ denote the set of frames with valid coordinates $(x, y)$. For any frame $f \notin \mathcal{V}$ with missing coordinates, the nearest valid preceding and succeeding frames are defined as

$$f_{\mathrm{p}} = \max\{\, t \in \mathcal{V} \mid t < f \,\}, \tag{8}$$
$$f_{\mathrm{n}} = \min\{\, t \in \mathcal{V} \mid t > f \,\}. \tag{9}$$

Missing coordinates are estimated according to the availability of $f_{\mathrm{p}}$ and $f_{\mathrm{n}}$ as

$$c(f) = \begin{cases} c(f_{\mathrm{p}}) + \dfrac{f - f_{\mathrm{p}}}{f_{\mathrm{n}} - f_{\mathrm{p}}} \\ \qquad \times \big(c(f_{\mathrm{n}}) - c(f_{\mathrm{p}})\big), & f_{\mathrm{p}}, f_{\mathrm{n}} \text{ exist}, \\ c(f_{\mathrm{p}}), & f_{\mathrm{p}} \text{ exists only}, \\ c(f_{\mathrm{n}}), & f_{\mathrm{n}} \text{ exists only}. \end{cases} \tag{10}$$

where, $c \in \{x, y\}$. The second case ($f_{\mathrm{p}}$ exists only) in Equation 10 assumes the insect remains stationary at its last known position. The third case ($f_{\mathrm{n}}$ exists only) assumes that the insect is already at its next known position until the missing frame is reached. However, this scenario is unlikely to occur in practice, as all tracks are initiated only after a valid insect detection.

After interpolation, the reconstructed trajectory is defined as

$$P = \{(f, x(f), y(f)) \mid f \in F\}. \tag{11}$$

#### 3.2.3 Software implementation

We extended Polytrack to incorporate frame preprocessing and trajectory reconstruction steps for compressed videos. This is available open-source on GitHub via the link provided under the Code Availability section. The deep learning-based insect detection model used in Polytrack is Ultralytics YOLOv11 (Jocher & Qiu, 2024), trained on a subset of the annotated dataset published in (Ratnayake, Amarathunga, Zaman, Dyer, & Dorin, 2022). Further details on the dataset and detection models are provided in the Supplementary Information.

### 3.3 Test datasets

To assess our video compression algorithm we used six real-world datasets of multiple videos encompassing a range of insect monitoring scenarios, application contexts, scene complexities, and recording modes (Table 1). Fig. 2 shows snapshots of application environments and Fig. 3 shows an analysis of inter-frame variations across the test datasets, highlighting changes caused by animal and insect movement, weather conditions, background disturbances, and illumination shifts. These factors result in both dense and sparse motion scenarios within each dataset.

## 4 Results

We conducted experiments evaluating performance of our algorithm to compress videos and its ability to retain information critical for automated tracking and manual insect behavioural analysis. In this section we present the results of these experiments. A detailed description of experimental results is available in Supplementary Information.

### 4.1 Video compression

We conducted two sets of experiments to evaluate the presented EcoMotionZip. The first experiment (Sec. 4.1.1) assessed file size reduction and frame count reduction achieved through EcoMotionZip for the test datasets. The second experiment (Sec. 4.1.2) compared its video compression performance with state-of-the-art codecs used in video compression that are commonly employed in camera traps.

#### 4.1.1 File size and frame count reduction

We evaluated video compression performance by comparing the reduction in file size and frame count. We preserved the original video characteristics, including resolution, frame rate, and codec, to enable a clear and controlled evaluation of our algorithm's compression performance. Results are shown in Table 2. This experiment was carried out on a Raspberry Pi 5 (8 GB) microcomputer.

The proposed algorithm achieved an average of 87% video compression across all test datasets. The minimum recorded compression was 71% for the Droissart et al. (2021) dataset. The compression ratio was particularly high for datasets where the environment remained relatively stationary during data collection, such as those in Ratnayake et al. (2022, 2020); van der Voort et al. (2022).

**Table 1**: **Test video dataset information.** "Application Environment" describes the monitoring environment and plant species recorded in the videos. "Recording Method" presents the video capture method. "Camera" describes the camera model used, "No. Videos" shows the number of videos in each dataset, "Video Codec" is the codec used, "Video Resol." is the video resolution, "FPS" is the recording frame rate.

| Dataset | Application Environment | Recording Method | Camera | No. Videos | Video Codec | Video Resol. | FPS |
|---|---|---|---|---|---|---|---|
| Naqvi et al. (2022) | Urban garden (Mixed native and exotic species) | Motion triggered / Time Lapse | Bushnell Nature-View HD 119740 | 3 | H264 | 1920, 1080 | 30 |
| Ratnayake et al. (2020) | Urban garden (Scaevola sp. native) | Continuous | Samsung Galaxy S8 | 7 | H264 | 1920, 1080 | 60 |
| Ratnayake et al. (2022) | Commercial farm (Fragaria × ananassa, Strawberry crop) | Continuous | Raspberry Pi V2 | 10 | MPEG4 | 1920, 1080 | 30 |
| van der Voort et al. (2022) | Controlled environment (Orchidaceae sp.) | Motion triggered | Raspberry Pi V2 | 1 | H264 | 1920, 1080 | 24 |
| Nest Monitoring | Bee nest (Amegilla sp.) | Continuous | Sony Handycam HDR-CX405 | 3 | H264 | 1920, 1080 | 25 |
| Droissart et al. (2021) | Semi-controlled environment (Multiple sp.) | Continuous | Raspberry Pi V2 | 3 | H264 | 1296, 972 | - |

* Droissart et al. (2021) dataset contains videos with different frame rates.

Notably, for all test videos, the file size reduction exceeded the frame count reduction. This difference was much more prominent for datasets recorded with motion-based triggers (Naqvi et al., 2022; van der Voort et al., 2022). This suggests that pixel-wise analysis of video frames, compared to simple motion detection, can achieve a higher compression rate (see Discussion).

### 4.1.2 Performance comparison with state-of-the-art video codecs

To benchmark the performance of our proposed video compression method against established standards, we evaluated it with three codec schemes: MJPEG (Motion JPEG) (Pennebaker & Mitchell, 1992), H.264/AVC (Wiegand et al., 2003), and H.265/HEVC (Sullivan et al., 2012). MJPEG was included as a legacy baseline, H.264 as the current industry standard used in camera-traps, and H.265 as the successor to H.264. Compression efficiency was assessed in terms of file size reduction and processing time. For each dataset, videos were first re-encoded independently using these codecs to establish baseline performance. Subsequently, the same codecs were employed as the encoder within the EcoMotionZip Writer component (Sec. 3.1.3), allowing us to measure the additional compression achieved when our method was integrated with each standard codec. The video resolution, frame rate, and bitrate were preserved as in the original recordings. MJPEG was stored in the AVI container, whereas H.264 and H.265 were packaged within the MP4 container. The experimental results are summarised in Table 3.

Across the six datasets, integrating the codecs into EcoMotionZip (Codec+EMZ) consistently improved compression efficiency compared to stand-alone encoding. Among the three codecs, H.265 achieved the strongest compression, yielding the smallest file sizes overall, followed by H.264 and MJPEG. The largest reduction in file size was observed with H.265+EMZ, which produced

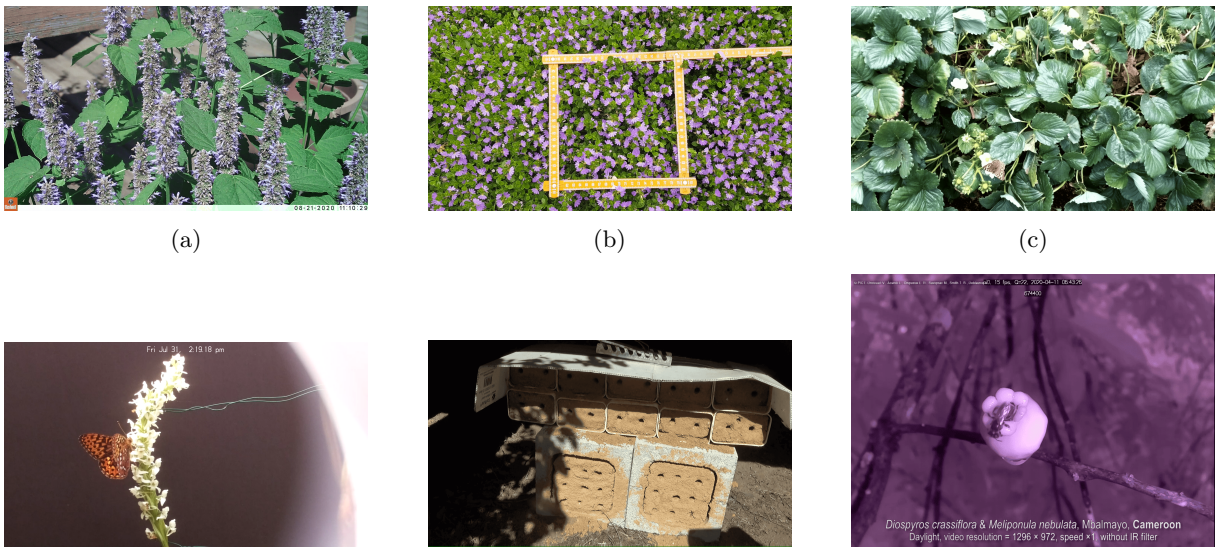



**Fig. 2**: **Application environments in the test dataset.** (a) Naqvi et al. (2022), (b) Ratnayake et al. (2020), (c) Ratnayake et al. (2022), (d) van der Voort et al. (2022), (e) Nest Monitoring, and (f) Droissart et al. (2021)

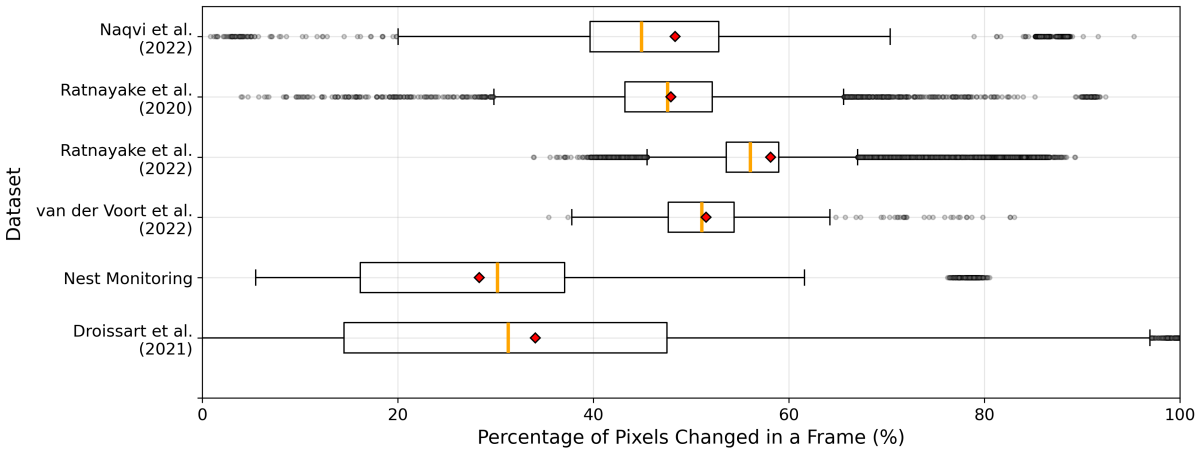
files that were on average 47.1% smaller than those generated by stand-alone H.265. This trend held across all datasets except van der Voort et al. (2022), where the stand-alone H.265 encoder produced a slightly smaller file than H.265+EMZ version (1.41 vs 1.60 MB; Table 3a). Throughput tests on a Raspberry Pi 5 (8 GB) demonstrated that H.264-based configurations were the fastest across all datasets, with EMZ integration further reducing processing times relative to their stand-alone counterparts (Table 3b). H.264+EMZ reduced average processing time by 56.07% compared to stand-alone H.264. On the Raspberry Pi 5, the stand-alone H.265 encoder failed to complete on Ratnayake et al. (2020) and the Nest Monitoring dataset, and the stand-alone MJPEG encoder failed on the Nest Monitoring dataset, with failures attributable to out-of-memory conditions triggered by high-frame-rate, large video files. Collectively, these results indicate that Eco-MotionZip reduces processing load while preserving and often enhancing compression effectiveness.

## 4.2 Behavioural analysis

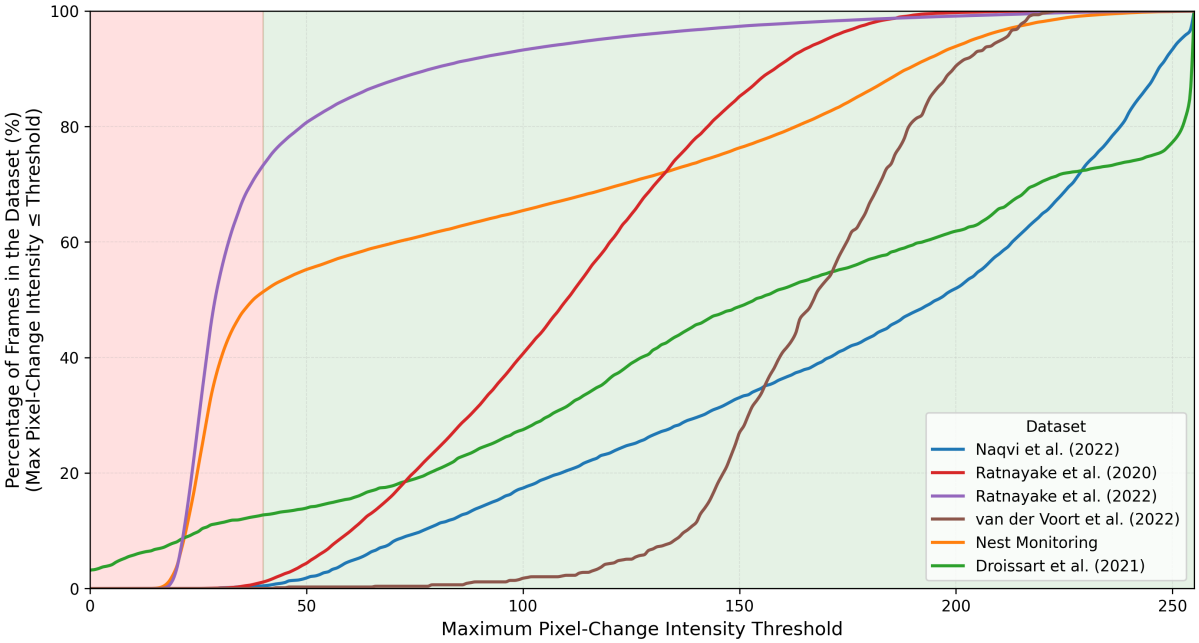
We evaluated our algorithm's ability to preserve animal behaviour information for our case study by comparing the number of insect appearances detected in raw and compressed videos using both automated and manual techniques. We used the dataset Ratnayake et al. (2022) for this experiment as it has the highest video compression and hence, plausibly, the highest likelihood of information loss. This dataset contains video of four insect types: honeybees, syrphid flies, lepidopterans, and vespids. We followed the procedure outlined in Ratnayake et al. (2023) to record insect events in this dataset, as described in Sections 4.2.1 and 4.2.2.

### 4.2.1 Automated video observations

In this experiment, we evaluated our compression algorithm's ability to retain insect motion and the extended Polytrack software (Sec. 3.2) to reconstruct spatiotemporal movement from compressed videos. For this evaluation, we processed the compressed test dataset using the extended Polytrack software to extract insect trajectories.




(a)




(b)

**Fig. 3**: **Analysis of inter-frame variations across datasets.** (a) Percentage of pixels changed per frame, defined as pixels whose intensity differs between consecutive frames. Red diamonds indicate means and orange lines indicate medians. (b) Empirical cumulative distribution function (ECDF) of the maximum per-frame pixel-change intensity for each dataset. The x-axis shows intensity thresholds (0 = no change; 255 = maximum change), and the y-axis shows the proportion of frames at or below each threshold. The red-shaded region denotes low-intensity changes discarded by *EcoMotionZip* (threshold $\leq 40$), while the green-shaded region indicates retained intensities.

The extracted tracks were then manually verified by by reviewing insect images associated with each saved track to correct insect type identifications and remove non-insect detections. Insect tracks that are less than five frames in length were

**Table 2**: **Results of the video compression.** "No. Frames" and "File Size (MB)" data for raw and processed videos show dataset total frame counts and file sizes. Reported file sizes for processed videos are the totals of compressed video and CSV files storing video supporting data. "Frame Reduc. (%)" and "File Size Reduc. (%)" show reduction in total frame count and test video file size. The original video codec was used for encoding in EcoMotionZip.

| Dataset | Raw Videos | | Processed Videos | | Frame Reduc. (%) | File Size Reduc. (%) |
|---|---|---|---|---|---|---|
| | No. Frames | File Size (MB) | No. Frames | File Size (MB) | | |
| Naqvi et al. (2022) | 5445 | 327.48 | 4166 | 70.61 | 23.49 | 78.44 |
| Ratnayake et al. (2020) | 22269 | 442.45 | 9989 | 39.48 | 55.14 | 91.08 |
| Ratnayake et al. (2022) | 179912 | 10895.05 | 12423 | 266.02 | 93.03 | 97.56 |
| van der Voort et al. (2022) | 790 | 23.59 | 775 | 2.47 | 1.90 | 89.53 |
| Nest Monitoring | 56664 | 1303.21 | 14331 | 52.92 | 74.71 | 95.94 |
| Droissart et al. (2021) | 5471 | 73.09 | 3862 | 20.80 | 29.41 | 71.54 |

also removed and were not analysed (Ratnayake et al., 2023). These tracks were then overlaid onto the uncompressed raw videos and manually reviewed frame-by-frame to compare detected positions against the actual position of the insect. We recorded True Positive (TP), False Positive (FP), and False Negative (FN) detections. A detection was classified as a True Positive if the insect's coordinates overlapped with its body area. If the coordinates were placed outside the insect's body area, it was categorised as a False Positive. If the insect's position was missing in any frame where it should have been recorded, it was counted as a False Negative. The recorded TP, FP, and FN values were then used to calculate Precision (Equation 12), Recall (Equation 13), and F-Score (Equation 14). These metrics were calculated for all ground-truth tracks documented in Ratnayake et al. (2023). The flower visit data recorded in the extracted tracks was used to evaluate the system's ability to retain information related to insect-flower interactions. If an insect's position was recorded on a flower for more than five frames, this was considered to be a flower visit event. Table 4 presents the results of automatic insect tracking in compressed videos compared to raw videos (Ratnayake et al., 2023). Fig. 4 shows the insect trajectories extracted from compressed videos. A detailed description of experimental results is available in Supplementary Information.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{12}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{13}$$

$$F\text{-}Score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{14}$$

The extended Polytrack algorithm successfully detected and tracked all honeybee appearances in the test videos (Table 4a). However, it produced more identity swaps in compressed videos compared to raw videos due to multiple tracks being generated for a single insect. Honeybee detection precision remained similar between raw and compressed videos, but the extended Polytrack detected 4% more bee positions in compressed videos, as indicated by an improved recall value of 0.96, which also resulted in a higher F-score. Tracking syrphids from compressed videos showed reduced performance compared to raw videos, with six additional tracks generated and precision

**Table 3**: **Results of (a) video compression efficiency and (b) processing time using MJPEG, H.264/AVC, and H.265/HEVC codecs.** For each codec, "Codec" denotes the baseline results from stand-alone encoding, while "Codec+EMZ" denotes the results when the codec was integrated as the encoder within EcoMotionZip Writer component. In (a), compression efficiency is reported in terms of file size (MB), and in (b), mean processing times (seconds, $Mean \pm Standard\ Deviation$) are presented from three replicates executed on a Raspberry Pi 5 (8 GB). The best-performing algorithm is indicated in bold.

(a)

| Dataset | MJPEG (MB) | | H.264 (MB) | | H.265 (MB) | |
|---|---|---|---|---|---|---|
| | Codec | Codec+EMZ | Codec | Codec+EMZ | Codec | Codec+EMZ |
| Naqvi et al. (2022) | 2275.23 | 574.76 | 190.96 | 70.30 | 73.54 | **44.69** |
| Ratnayake et al. (2020) | 13092.79 | 412.83 | 256.62 | 39.12 | 81.30 | **29.92** |
| Ratnayake et al. (2022) | 48332.21 | 480.91 | 3142.51 | 60.53 | 627.97 | **37.83** |
| van der Voort et al. (2022) | 76.74 | 26.49 | 9.38 | 2.43 | **1.41** | 1.60 |
| Nest Monitoring | 16420.09 | 539.71 | 925.60 | 52.14 | 200.35 | **39.15** |
| Droissart et al. (2021) | 565.21 | 142.81 | 43.63 | 20.77 | 15.63 | **12.63** |

(b)

| Dataset | MJPEG (sec) | | H.264 (sec) | | H.265 (sec) | |
|---|---|---|---|---|---|---|
| | Codec | Codec+EMZ | Codec | Codec+EMZ | Codec | Codec+EMZ |
| Naqvi et al. (2022) | $481.61 \pm 0.82$ | $305.74 \pm 3.05$ | $344.82 \pm 0.55$ | $\mathbf{209.07 \pm 0.81}$ | $1469.76 \pm 0.6$ | $1020.67 \pm 0.35$ |
| Ratnayake et al. (2020) | $2154.56 \pm 2.53$ | $683.46 \pm 1.79$ | $1015.35 \pm 1.26$ | $\mathbf{395.14 \pm 3.23}$ | NA* | $1649.5 \pm 2.06$ |
| Ratnayake et al. (2022) | $14142.59 \pm 3.82$ | $1478.17 \pm 2.19$ | $9038.91 \pm 4.01$ | $\mathbf{1457.16 \pm 6.79}$ | $39856.5 \pm 19.81$ | $2976.57 \pm 2.83$ |
| van der Voort et al. (2022) | $55.57 \pm 0.02$ | $51.39 \pm 0.3$ | $46.12 \pm 0.02$ | $\mathbf{24.92 \pm 0.27}$ | $160.67 \pm 0.31$ | $124.82 \pm 0.09$ |
| Nest Monitoring | NA* | $1047.46 \pm 1.58$ | $2638.18 \pm 1.7$ | $\mathbf{784.87 \pm 2.29}$ | NA* | $2417.55 \pm 14.99$ |
| Droissart et al. (2021) | $249.7 \pm 0.32$ | $165.0 \pm 0.1$ | $189.09 \pm 0.37$ | $\mathbf{121.3 \pm 0.36}$ | $894.42 \pm 0.13$ | $599.25 \pm 0.12$ |

* Raspberry Pi 5 (8GB) failed to process three replicates of the dataset.

and recall dropping by 3% and 35%, respectively (see Discussion). However, compared to raw videos, all syrphid appearances were detected in the compressed videos. For Lepidoptera, tracking results were similar in both raw and compressed videos, with comparable detection evaluation metrics. Both methods failed to detect motion for one insect. In the case of vespids, the extended Polytrack missed detecting one insect in the compressed videos compared to raw videos and incorrectly merged two separate tracks into one. Detection precision for vespids decreased by 15% in compressed videos, mainly due to the trajectory reconstruction algorithm inaccurately estimating positions using linear interpolation. However, with trajectory reconstruction, the recall value improved by 15% compared to raw videos. In the flower visit data comparison (Table 4a), both compressed and raw videos missed two honeybee flower visits due to undetected flowers. However, for honeybees, syrphids, and Lepidoptera, processing compressed videos resulted in a higher number of false positive visits, primarily due to track segmentation errors (see Discussion). Additionally, the compressed videos failed to detect two flower visits by Lepidoptera, whereas all visits were successfully detected in the raw videos. For vespids, neither method detected any flower visits. Overall, in EcoMotionZip-compressed videos, the extended Polytrack detected and tracked 97.5% of the observed tracks (ground truth) and accurately identified 93.6% flower visits. Except for syrphids, all other insect groups were tracked within a 2% difference in F-score compared to raw, uncompressed videos.

### 4.2.2 Manual video observations

We conducted a frame-by-frame analysis of the compressed video footage and manually counted

**Table 4: Comparison of automated (a) insect tracking, and (b) flower visit monitoring results for the Ratnayake et al. (2022) test video dataset using raw and compressed videos.** Results for raw, uncompressed videos are adopted from Ratnayake et al. (2023) and shown under "Raw", while results for compressed videos are shown under "Com". In (a), "No. of Obs." represents the number of insect observations recorded through human analysis of raw video. "Trackletts Generated" indicates the number of tracklets extracted by Polytrack. "Track Evaluation" categorises the tracklets as True Positives, False Negatives, or False Positives/Identity Swaps, with multiple tracks generated for a single insect counted as False Positives/Identity Swaps. "Detection Evaluation" presents the average precision, recall, and F-score metrics for tracked insects. In (b), "Observed Visits" shows the total number of insect visits to flowers counted through human observations of raw videos. "Flower Visit Evaluation" shows the evaluation of insect flower visits automatically identified through automated analysis of videos. A detailed description of tracking results for compressed videos is provided in the Supplementary Information.

(a)

| Insect Type | No. of Obs. | Track Evaluation | | | | | | | | Detection Evaluation | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Trackletts Generated | | True Positive | | False Negative | | False Pos./ Identity Swap | | Precision | | Recall | | F-score | |
| | | Raw | Com | Raw | Com | Raw | Com | Raw | Com | Raw | Com | Raw | Com | Raw | Com |
| Honeybee | 20 | 23 | 26 | 20 | 20 | 0 | 0 | 3 | 6 | 0.99 | 0.99 | 0.92 | 0.96 | 0.95 | 0.97 |
| Syrphidae | 6 | 6 | 13 | 5 | 6 | 1 | 0 | 1 | 7 | 1.00 | 0.97 | 0.71 | 0.46 | 0.81 | 0.55 |
| Lepidoptera | 4 | 5 | 5 | 3 | 3 | 1 | 1 | 2 | 2 | 0.99 | 1.00 | 0.71 | 0.71 | 0.81 | 0.80 |
| Vespidae | 10 | 10 | 10 | 10 | 9 | 0 | 1 | 0 | 1* | 1.00 | 0.85 | 0.73 | 0.84 | 0.83 | 0.82 |

* Includes merged tracks.

(b)

| Insect Type | Observed Visits | Flower Visit Evaluation | | | | | |
|---|---|---|---|---|---|---|---|
| | | True Positive | | False Negative | | False Positive | |
| | | Raw | Com | Raw | Com | Raw | Com |
| Honeybee | 67 | 65 | 65 | 2 | 2 | 0 | 1 |
| Syrphidae | 5 | 4 | 4 | 1 | 1 | 1 | 3 |
| Lepidoptera | 6 | 6 | 4 | 0 | 2 | 1 | 2 |
| Vespidae | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

the number of insects, categorised them by type, and recorded the number of flowers visited by each insect. When an insect first appeared in the video, we played the footage frame-by-frame to analyse its movement. If an insect exited the frame but subsequently reappeared, or if it was occluded by foliage and re-emerged, we counted it as a new individual. Finally, when an insect landed on a fully visible and completely open flower, or if the compressed video had a gap in recorded frames when an insect was over a flower, we considered and recorded this as a flower visit. We only considered insects that appeared for more than 5 frames for the analysis, following the procedure outlined in Ratnayake et al. (2023). We compared the results of these observations against the visual observation results published in the study. Results are shown in Table 5.

The compressed video preserved all insect count information. These insects, primarily small-bodied members of the Syrphidae, Lepidoptera, and Vespidae families, often visually blend into the environment, leading to them being missed by humans in unprocessed, raw video sequences where they are viewed against complex backgrounds (see Discussion). In comparison, the appearances of these insects were noticeable due to colour highlights against blacked-out surroundings in the compressed video. The compressed video captured all flower visits except for those by

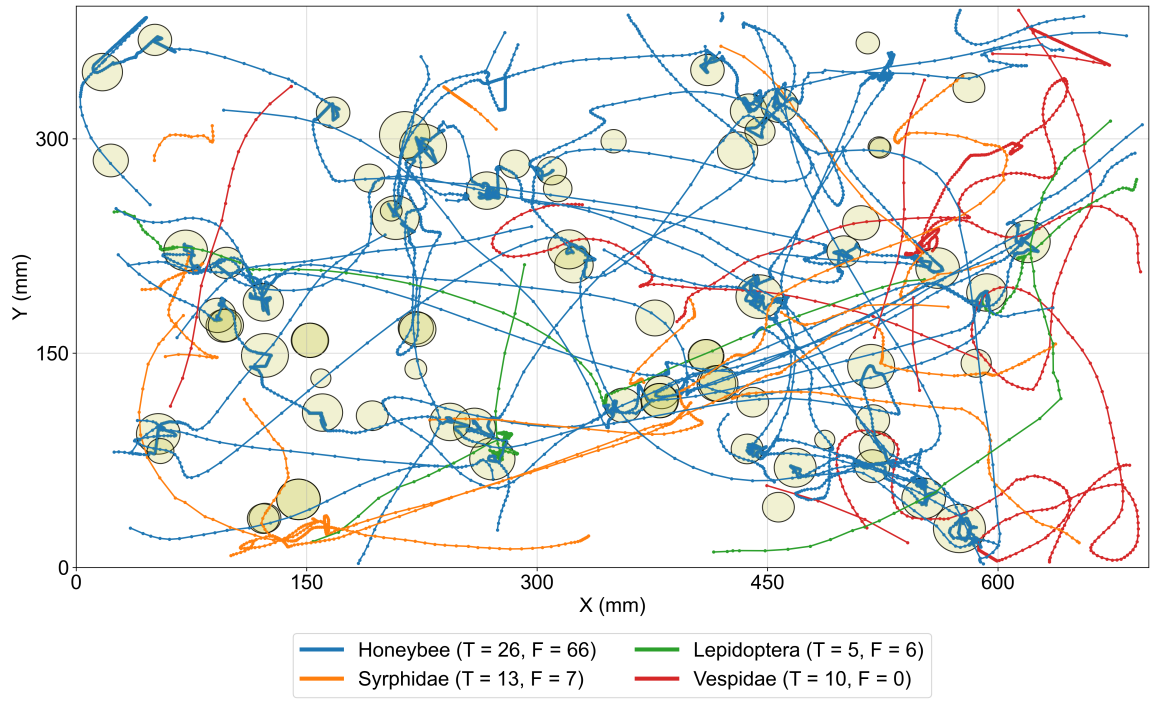



**Fig. 4: Insect trajectories and flower positions extracted using Extended Polytrack from EcoMotionZip for compressed Ratnayake et al. (2022) dataset videos.** In the legend, for each type of insect, "T" indicates the number of tracks recorded, and "F" denotes the number of flower visits observed. Flower locations are highlighted with yellow circles.

honeybees. This discrepancy occurred because the compression algorithm discarded frames with no motion resulting in missing frames where insects, particularly honeybees, decelerate rapidly and hover before landing on a flower, causing their motion to appear zero.

## 4.3 Edge-device performance

We evaluated the performance of the proposed video compression algorithm on three edge computing platforms used for camera traps and insect pollinator monitoring: general purpose Raspberry Pi models 4 and 5 (8 GB), and a Nvidia Jetson Nano (2 GB). A detailed devices specification is provided in Supplementary information.

Processing time was calculated by measuring the time taken by each platform to continuously process each video dataset. A USB power meter connected to the devices measured the energy consumed by each processing task. Experiments were conducted in a temperature-controlled room with the temperature varying between $20 \circ C - 23 \circ C$ Celsius. The power consumed by the video processing task was determined by removing the idle power consumption of the device from the power consumption measured by the USB power meter. Experiments were conducted for three replicates. Results are shown in Table 6 and Fig. 5.

The Raspberry Pi 5 outperformed the other two platforms on most test video datasets, achieving faster processing speeds and shorter processing times. Notably, the Jetson Nano was faster for the specific case of the Droissart et al. (2021) dataset. Raspberry Pi 4 displayed slow processing speeds for all datasets. Jetson Nano was power efficient for processing smaller video datasets. However, when handling larger datasets like Ratnayake et al. (2022) the Raspberry Pi 4 was the most energy-efficient option.

**Table 5**: **Comparison of manual insect counts and flower visits in raw / compressed videos.** "Raw Video" presents observations from raw videos of Ratnayake et al. (2022). "Comp. video" presents observations from our compressed videos. "Insects Missed" and "Visits Missed" count insect appearances and flower visits unobserved in compressed videos but counted from raw video observations. "New Insects Observed" and "New Visits Observed" show insect appearances and flower visits observed in compressed videos but unrecorded in observations made on the raw video dataset. Counts related to "New Insects Observed" and "New Visits Observed" are included in "Comp. video" recordings.

| Insect type | Insect Counts | | | | No. of flower visits | | | |
|---|---|---|---|---|---|---|---|---|
| | Raw video | Comp. video | Insects missed | New insects observed | Raw video | Comp. video | Visits missed | New visits observed |
| Honeybee | 20 | 20 | 0 | 0 | 67 | 67 | 2 | 2 |
| Syrphidae | 6 | 9 | 0 | 3 | 5 | 6 | 0 | 1 |
| Lepidoptera | 4 | 5 | 0 | 1 | 6 | 6 | 0 | 0 |
| Vespidae | 10 | 13 | 0 | 3 | 0 | 0 | 0 | 0 |

**Table 6**: **Processing time for test video dataset on edge computing platforms.** "Dur. (sec)" and "FPS" show the total video duration and average video frame-rate for raw videos. "Time (sec)" and "Speed (fps)" present the processing time (Mean ± Standard deviation) and the average number of frames processed per second on each platform, based on three replicates.

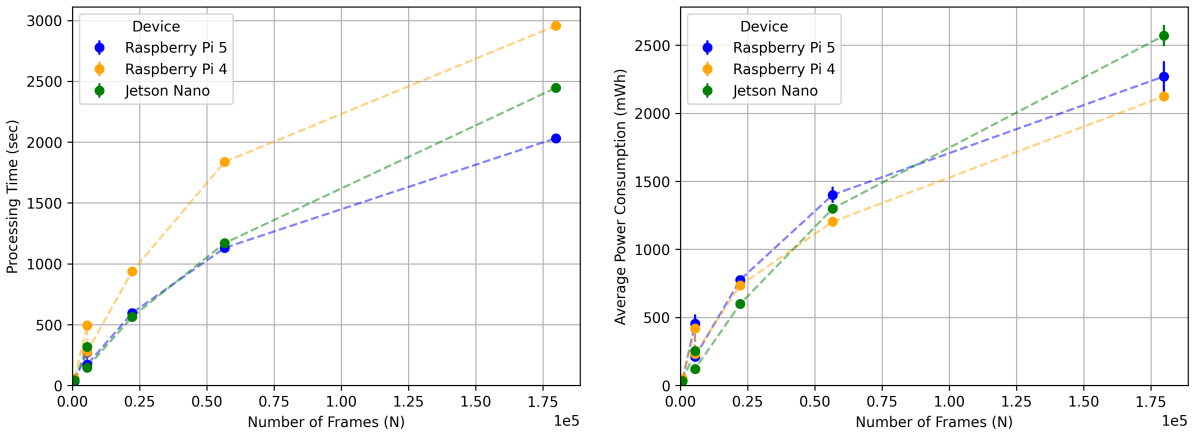
| Dataset | Raw Videos | | Raspberry Pi 5 | | Raspberry Pi 4 | | Jetson Nano | |
|---|---|---|---|---|---|---|---|---|
| | Dur. (sec) | FPS | Time (sec) | Speed (fps) | Time (sec) | Speed (fps) | Time (sec) | Speed (fps) |
| Naqvi et al. (2022) | 181 | 30 | $272.1 \pm 0.7$ | 20.1 | $493.7 \pm 2.1$ | 11.0 | $316.4 \pm 6.2$ | 17.2 |
| Ratnayake et al. (2020) | 371 | 60 | $596.1 \pm 0.7$ | 37.4 | $938.4 \pm 1.2$ | 23.7 | $564.3 \pm 0.9$ | 39.5 |
| Ratnayake et al. (2022) | 5597 | 30 | $2032.5 \pm 6.1$ | 88.5 | $2956.6 \pm 7.0$ | 60.9 | $2446.6 \pm 1.7$ | 73.5 |
| van der Voort et al. (2022) | 33 | 24 | $35.5 \pm 0.3$ | 22.3 | $57.7 \pm 0.0$ | 13.7 | $41.7 \pm 0.0$ | 19.0 |
| Nest Monitoring | 2266 | 25 | $1131.2 \pm 6.9$ | 50.1 | $1837.0 \pm 5.1$ | 30.9 | $1169.6 \pm 3.0$ | 48.5 |
| Droissart et al. (2021) | 307 | 18 | $170.6 \pm 0.5$ | 32.1 | $277.0 \pm 0.7$ | 19.8 | $147.6 \pm 0.4$ | 37.0 |

# 5 Discussion

## 5.1 Video compression

Our algorithm achieved over 87% compression of insect monitoring videos across diverse monitoring environments (Table 2), while preserving key information (Table 5). This video compression performance exceeded that of widely used state-of-the-art stand-alone video codecs, both in video size reduction (Table 3a) and in processing speed on edge devices (Table 3b). This compression performance translates to reduced storage and bandwidth needs, particularly beneficial for resource-constrained edge-camera traps. The algorithm achieved file size reductions exceeding frame count reductions across all datasets. This was particularly pronounced in controlled environments with minimal background changes and for video files recorded with motion-based triggers or containing motion data in the majority of recorded video frames (e.g Droissart et al. (2021); Naqvi et al. (2022); van der Voort et al. (2022), Fig. 3 and Table 2). In these cases, our pixel-wise motion analysis selectively eliminated data from non-moving pixels while retaining essential information from pixels with motion for further analysis. This demonstrates the adaptability of our approach in compressing videos, regardless of the environmental conditions, the number of animals in the frame, or the type of trigger used for recording.

Combining our approach with state-of-the-art video codecs enabled video compression beyond what is achievable with stand-alone codecs




(a) (b)

**Fig. 5**: **Performance of the proposed algorithm on edge processing devices.** (a) Average Processing Time and (b) Average Power Consumption of the proposed algorithm on various edge device platforms when processing the test datasets. Each dataset was processed in three replicates. x-axis shows the number of frames in the test dataset and error bars show the standard deviation of the mean.

(Table 3a). This was accomplished by integrating the codec into the Writer component of the EcoMotionZip algorithm (Sec. 3.1.3). The reduction in file size was most pronounced in datasets with a high degree of frame reduction (e.g., Ratnayake et al. (2022), Nest Monitoring; Table 2), and in those with a higher proportion of frames falling below the maximum pixel-change intensity threshold (Fig. 3b). These results indicate that file size reduction was achieved primarily through the selective discarding of frames and pixel regions with minimal inter-frame motion. In addition to improved compression efficiency, our algorithm processed datasets in less time than stand-alone codecs (Table 3b). This was largely due to the elimination of unnecessary write operations for frames and pixels lacking valuable insect behavioural information, thereby reducing computational overhead across all datasets and codecs. Notably, the Raspberry Pi 5 (8 GB) was unable to process datasets such as Ratnayake et al. (2020) and Nest Monitoring using the stand-alone H.265 codec, due to out-of-memory errors. This limitation further highlights the advantages of our integrated approach for resource-constrained edge devices.

Our approach utilises foreground changes as motion triggers without assessing cause of motion. While this approach can record animals with few false positives (Table 5), compressed videos did contain false (non-insect) detections caused by wind and illumination changes. These add to the compressed video file size, especially in dynamic monitoring environments. One potential strategy to minimise false detections involves assessing foreground change-based motion triggers using deep learning prior to storing the video frames (Bjerge, Mann, & Høye, 2021; Sittinger, Uhler, Pink, & Herz, 2023; Tan et al., 2022; Zualkernan et al., 2022). While this may improve compression, the effectiveness of the camera trap then hinges on the accuracy of the deep learning which, if inadequately trained, may miss events (Tan et al., 2022). This is an issue especially for scarce or undocumented animal species that might be extremely valuable to record (van Klink et al., 2022). In addition, deep learning demands greater computational resources and more specialised, relatively expensive hardware than our approach, for it to process images on a device (Sittinger et al., 2023). Deep learning may also increase frame processing latency, reducing capture rates and increasing the likelihood of missing fast animals. The efficacy of camera traps relies on their ability to capture a comprehensive representation of animals and their environment. Unforeseen

animal interactions, new species, and previously undocumented behaviours may be overlooked if the camera is only programmed to capture data based on existing information. Therefore, it can be crucial to collect animal data unaligned with past (training) data. Furthermore, the complexities associated with animal behaviour may surpass the capabilities of current artificial intelligence models that can only be trained to identify a selected number of behaviours (Arablouei et al., 2023). A complete raw record of animal appearances and behaviours though, allows experts to analyse videos beyond AI's current capability. The emergence of zero-shot object detection models shows promise in addressing the limitations of existing AI models (Bansal, Sikka, Sharma, Chellappa, & Divakaran, 2018). Therefore, future research on their application in camera trap-based animal studies would be valuable.

## 5.2 Behavioural analysis

Increasingly, automation of animal behaviour studies (Marks et al., 2022; Schindler & Steinhage, 2021), including outdoor insect tracking (Gebauer, Thiele, Ouvrard, Sicard, & Risse, 2024; Ratnayake et al., 2023, 2021a; Sittinger et al., 2024), has enhanced our understanding of animal behaviour. Algorithms can independently process hours of video data, extracting animal movements and environmental interactions with little human intervention.

Polytrack, incorporating new preprocessing and trajectory reconstruction steps, successfully extracted 97.5% of insect tracks (Table 4a) and 93.6% of flower visits (Table 4b) from compressed videos, compared to those extracted from raw videos (Fig. 4). This demonstrates the feasibility of the extended Polytrack algorithm for automated tracking and behavioural analysis using compressed videos and the compressed videos to retain information on animal movement and behaviour for automated analysis. However, some tracks, particularly those of syrphids, were fragmented into segments during processing. This occurred primarily because these insects often remain stationary on flowers for extended periods. As a result, EcoMotionZip classified them as part of the background. When they resumed movement, the extended Polytrack algorithm identified them as new appearances, generating separate tracklets while the stationary period was recorded as a false negative detection. In addition, this issue led to double counting in flower visitation data, causing the number of visits extracted from compressed videos to be higher than those obtained from raw videos. Reduced tracking performance for insects such as syrphids has also been previously reported, primarily due to their relatively small size, which makes accurate detection with deep learning models challenging. Future research aimed at improving tracking methods for temporally inconsistent spatial data would be beneficial. Furthermore, advancements in individual animal identification will enhance tracking performance by linking separate tracks on either side of occlusions or stationary periods. Ultimately these improvements will lead to more robust and reliable animal tracking with compressed videos.

In our manual comparisons of compressed and raw videos, videos processed by the algorithm successfully retained all information required for counting insects (Table 5). Additionally, our human observations of compressed videos revealed more fast-moving vespids, relatively small syrphids, and Lepidoptera than did the raw video observations. By discarding frames and pixels lacking motion activity and preserving colour information in pixels with motion, our algorithm removes the distraction of complex images where insects are only a tiny proportion of visible pixels, allowing humans to accurately identify insects. Conversely, video playback observations with raw videos require careful study of the entire complex frame for extended periods, often with little or no insect activity visible. Human attention limitations (Simons & Chabris, 1999), fatigue (Faber, Maurits, & Lorist, 2012; Zett, Stratford, & Weise, 2022), and even low skill levels (Zett et al., 2022) then led to missed insect-related events. Discarding entire frames without motion activity clearly reduces the time, likelihood of errors, and ultimately the costs of ecological video observation (Breeze et al., 2021).

## 5.3 Edge device performance

The edge device performance evaluation (Table 6) shows processing speeds over 20 fps on all datasets on the Raspberry Pi 5. This was achieved by multi-threading processes for reading, processing, and writing video frames. Processing

speed increased in semi-controlled environments with sparse insect appearances (Ratnayake et al. (2022) and Nest Monitoring datasets), regardless of the percentage of pixel changes per frame (Fig.3). This indicates that our algorithm has the potential for real-time video recording and compression. However, the algorithm's performance depends on the application environment, weather and device specifications. With increased insect activity, the camera-trap hardware might reach its computational limits, resulting in frame drops and missed fast-moving insects. Therefore, while real-time video recording is feasible in some scenarios, compressing videos through post-processing remains more reliable if it is possible. We anticipate that the development of more energy-efficient and powerful hardware will eventually enable the real-time implementation of our method across broader monitoring conditions, as evidenced by the increase in processing speeds from Raspberry Pi model 4 to 5 (Table 6 and Fig. Fig.5a).

As dataset size increased, the compression algorithm consumed more power across all edge device platforms (Fig.5b). This adds to the overall monitor energy usage that includes video recording energy consumption. However, video compression can subsequently reduce the energy consumption and requirements associated with video transmission and storage by decreasing file sizes (Table 2).This is particularly beneficial for autonomous camera traps deployed in remote or poorly connected locations, as it lowers deployment costs and reduces the frequency of service visits required to replace batteries and storage media. The compression algorithm's value will vary depending on the relative availability of energy, storage, and transmission resources in a scenario. Its additional power consumption might be met by solar power generation for instance in some circumstances. Advances in energy-efficient edge hardware may also lower power consumption, further expanding its valuable range.

# 6 Conclusions

This paper presented a motion-analysis algorithm for compressing data captured by resource-constrained video camera traps. By analysing video frames pixel-by-pixel, our algorithm achieved an average compression of 87% on diverse test datasets while preserving all information crucial for animal behaviour analysis. When combined with the H.265/HEVC codec, compression improved by an additional 47.1% relative to stand-alone H.265/HEVC. These substantial file-size reductions significantly enhances the monitoring capabilities of remote camera traps by maximising monitoring duration, minimising storage requirements and bandwidth demands, and facilitating efficient data transfer from resource-limited devices. In addition, we introduced an extension to the existing open-source insect tracking software Polytrack, incorporating a preprocessing and trajectory reconstruction method. This enabled Polytrack's compatibility with compressed videos for automated insect tracking and behavioural analysis. The extended Polytrack successfully detected and tracked 97.5% of insects and accurately identified 93.6% of their flower visits from compressed videos, achieving accuracy levels comparable to those of uncompressed videos. These results demonstrate the effectiveness of the proposed system in facilitating automated behavioural analysis, offering a reliable solution for studying insect movement and interactions using compressed video data. In addition, our compression algorithm improved human observer effectiveness when extracting ecological data from videos. It directed observer attention to video sequences and frame regions with animal activity by removing sequences and pixels lacking relevant information. This alleviates human strain and fatigue associated with lengthy observation sessions. These reductions in camera trap resource requirements and observer time commitment have the potential to significantly reduce the cost of ecological monitoring. Hence, deploying this solution on camera traps can significantly advance ecological monitoring by improving the capabilities of existing systems.

**Supplementary information.** Additional and detailed experimental results on presented methods are available in the Supplementary Information file "Supplementary_Information.pdf".

**Acknowledgements.** Authors acknowledge Professor James Cook and his research team for sharing the videos of Nest Monitoring for experiments.

## Declarations

- Funding: This research was supported partially by the Australian Government through the ARC's Linkage Projects funding scheme (project LP210200213). Lex Gallon was supported by a scholarship from the Dept. of Data Science and AI, Faculty of Information Technology, Monash University.
- Conflict of interest: The authors have no competing interests to declare that are relevant to the content of this article.
- Ethics approval and consent to participate: Not applicable.
- Consent for publication: Not applicable.

  Data availability: Publicly accessible datasets analysed in this study are listed and linked in the Supplementary Information. Data derived from the analysis of source videos will be deposited in a permanent public repository (Figshare) and assigned a DOI upon acceptance of the manuscript. In the interim, these files are temporarily available at Google Drive. Video files generated during the analysis are available from the corresponding author.
- Code availability:
  - EcoMotionZip: github.com/malikaratnayake/EcoMotionZip
  - Polytrack: github.com/malikaratnayake/Polytrack
- Authors' contributions: Conceptualization: Malika Nisal Ratnayake, Lex Gallon, Adel N Toosi, Alan Dorin; Data curation: Malika Nisal Ratnayake, Lex Gallon; Formal analysis: Malika Nisal Ratnayake, Lex Gallon; Funding acquisition: Alan Dorin; Investigation: Malika Nisal Ratnayake, Lex Gallon, Adel N Toosi Alan Dorin; Methodology: Malika Nisal Ratnayake, Lex Gallon, Adel N Toosi Alan Dorin; Project administration: Adel N Toosi, Alan Dorin; Resources: Adel N Toosi, Alan Dorin; Software: Malika Nisal Ratnayake, Lex Gallon; Supervision: Malika Nisal Ratnayake, Adel N Toosi, Alan Dorin; Validation: Malika Nisal Ratnayake, Lex Gallon; Writing – original draft: Malika Nisal Ratnayake; Writing – review & editing: Malika Nisal Ratnayake, Lex Gallon, Adel N Toosi, Alan Dorin.

## References


Ahmed, Z., Hussain, A.J., Khan, W., Baker, T., Al-Askar, H., Lunn, J., ... Liatsis, P. (2020). Lossy and lossless video frame compression: A novel approach for high-temporal video data analytics. *Remote Sensing*, *12*(6), 1004,

Arablouei, R., Wang, L., Currie, L., Yates, J., Alvarenga, F.A., Bishop-Hurley, G.J. (2023). Animal behavior classification via deep learning on embedded systems. *Computers and Electronics in Agriculture*, *207*, 107707,

Bansal, A., Sikka, K., Sharma, G., Chellappa, R., Divakaran, A. (2018). Zero-shot object detection. *Proceedings of the european conference on computer vision (eccv)* (pp. 384–400).

Bjerge, K., Frigaard, C.E., Karstoft, H. (2023). Object detection of small insects in time-lapse camera recordings. *Sensors*, *23*(16), 7242,

Bjerge, K., Mann, H.M., Høye, T.T. (2021). Real-time insect tracking and monitoring with computer vision and deep learning. *Remote Sensing in Ecology and Conservation*, ,

Breeze, T.D., Bailey, A.P., Balcombe, K.G., Brereton, T., Comont, R., Edwards, M., ... others (2021). Pollinator monitoring more than pays for itself. *Journal of Applied Ecology*, *58*(1), 44–57,

Caravaggi, A., Banks, P.B., Burton, A.C., Finlay, C.M., Haswell, P.M., Hayward, M.W., ... Wood, M.D. (2017). A review of camera trapping for conservation behaviour research. *Remote Sensing in Ecology and Conservation*, *3*(3), 109–122,

Collett, R.A., & Fisher, D.O. (2017). Time-lapse camera trapping as an alternative to pitfall trapping for estimating activity of leaf litter arthropods. *Ecology and Evolution*, *7*(18), 7527–7533,

Dharmarathne, S.C., Jayasekara, E., Mahaulpatha, D., de Silva, K. (2022). Camera trap data reveals the habitat use and activity patterns of a secretive forest bird, sri lanka spurfowl galloperdix bicalcarata. *Journal of Wildlife and Biodiversity*, *6*(Suppl. 1), 100–118,

Dou, X., Cao, X., Zhang, X. (2024). Region-of-interest based coding scheme for live videos. *Applied Sciences*, *14*(9), 3823,

Droissart, V., Azandi, L., Onguene, E.R., Savignac, M., Smith, T.B., Deblauwe, V. (2021). Pict: A low-cost, modular, open-source camera trap system to study plant–insect interactions. *Methods in Ecology and Evolution*, *12*(8), 1389–1396,

Faber, L.G., Maurits, N.M., Lorist, M.M. (2012). Mental fatigue affects visual selective attention. *PloS one*, *7*(10), e48073,

Fairhurst, G., Nazir, S., Verdicchio, F. (2013). Sensor-based digital platform for remote environmental monitoring: National telford institute (wireless sensor networking and its engineering applications). *Wireless sensor networking and its engineering applications: New directions in measurement and control.*

Feng, J., Sun, Y., Li, H., Xiao, Y., Zhang, D., Smith, J.L., ... Wang, T. (2021). Assessing mammal species richness and occupancy in a northeast asian temperate forest shared by cattle. *Diversity and Distributions*, *27*(5), 857–872,

Food & Agriculture Organization (2019). Global action on pollination services for sustainable agriculture.

Gazzea, E., Batáry, P., Marini, L. (2023). Global meta-analysis shows reduced quality of food crops under inadequate animal pollination. *Nature communications*, *14*(1), 4463,

Gebauer, E., Thiele, S., Ouvrard, P., Sicard, A., Risse, B. (2024). Towards a dynamic vision sensor-based insect camera trap. *Proceedings of the ieee/cvf winter conference on applications of computer vision* (pp. 7157–7166).

Green, S.E., Stephens, P.A., Whittingham, M.J., Hill, R.A. (2023). Camera trapping with photos and videos: implications for ecology and citizen science. *Remote Sensing in Ecology and Conservation*, *9*(2), 268–283,

Head, J.S., Robbins, M.M., Mundry, R., Makaga, L., Boesch, C. (2012). Remote video-camera traps measure habitat use and competitive exclusion among sympatric chimpanzee, gorilla and elephant in loango national park, gabon. *Journal of Tropical Ecology*, *28*(6), 571–583,

Janisch, J., Mitoyen, C., Perinot, E., Spezie, G., Fusani, L., Quigley, C. (2021). Video recording and analysis of avian movements and behavior: insights from courtship case studies. *Integrative and Comparative Biology*, *61*(4), 1378–1393,

Jocher, G., & Qiu, J. (2024). *Ultralytics yolo11.* Retrieved from https://github.com/ultralytics/ultralytics

Krauss, S.L., Roberts, D.G., Phillips, R.D., Edwards, C. (2018). Effectiveness of camera traps for quantifying daytime and nighttime visitation by vertebrate pollinators. *Ecology and Evolution*, *8*(18), 9304–9314,

Le, H., Zhang, L., Said, A., Sautiere, G., Yang, Y., Shrestha, P., ... Wiggers, A. (2022). Mobilecodec: neural inter-frame video compression on mobile devices. *Proceedings of the 13th acm multimedia systems conference* (pp. 324–330).

Liu, D., Li, Y., Lin, J., Li, H., Wu, F. (2020). Deep learning-based video coding: A review and a case study. *ACM Computing Surveys (CSUR)*, *53*(1), 1–35,

Lovell, C., Li, S., Turner, J., Carbone, C. (2022). The effect of habitat and human disturbance on the spatiotemporal activity of two urban carnivores: The results of an intensive camera trap study. *Ecology and evolution*, *12*(3), e8746,

Marks, M., Jin, Q., Sturman, O., von Ziegler, L., Kollmorgen, S., von der Behrens, W., ... Yanik, M.F. (2022). Deep-learning-based identification, tracking, pose estimation and behaviour classification of interacting primates and mice in complex environments. *Nature machine intelligence*, *4*(4), 331–340,

Melidonis, C.A., & Peter, C.I. (2015). Diurnal pollination, primarily by a single species of rodent, documented in protea foliosa using modified camera traps. *South African Journal of Botany*, *97*, 9–15,

Naqvi, Q., Wolff, P.J., Molano-Flores, B., Sperry, J.H. (2022). Camera traps are an effective tool for monitoring insect–plant interactions. *Ecology and Evolution*, *12*(6), e8962,

Nykänen, M., Pöysä, H., Matala, J., Kunnasranta, M. (2023). Motion detection or time lapse? a comparison of camera trap triggers in the monitoring of elusive ground dwelling birds.

Ortmann, C., & Johnson, S. (2021). How reliable are motion-triggered camera traps for detecting small mammals and birds in ecological studies? *Journal of Zoology*, *313*(3), 202–207,

Peleg, S., & Pritch, Y. (2012, November 13). *Method and system for video indexing and video synopsis* (No. US8311277B2). Retrieved from https://patents.google.com/patent/US8311277B2/en

Pennebaker, W.B., & Mitchell, J.L. (1992). *Jpeg: Still image data compression standard*. Springer Science & Business Media.

Perugachi-Diaz, Y., Sautière, G., Abati, D., Yang, Y., Habibian, A., Cohen, T.S. (2022). Region-of-interest based neural video compression. *arXiv preprint arXiv:2203.01978*, ,

Rampim, L.E., Sartorello, L.R., Fragoso, C.E., Haberfeld, M., Devlin, A.L. (2020). Antagonistic interactions between predator and prey: mobbing of jaguars (panthera onca) by white-lipped peccaries (tayassu pecari). *acta ethologica*, *23*(1), 45–48,

Ratnayake, M.N., Amarathunga, D.C., Zaman, A., Dyer, A.G., Dorin, A. (2022, 11). *Spatial Monitoring and Insect Behavioural Analysis Dataset.*

Ratnayake, M.N., Amarathunga, D.C., Zaman, A., Dyer, A.G., Dorin, A. (2023). Spatial monitoring and insect behavioural analysis using computer vision for precision pollination. *International Journal of Computer Vision*, *131*(3), 591–606,

Ratnayake, M.N., Dyer, A., Dorin, A. (2020, 8). *Honeybee video tracking data.*

Ratnayake, M.N., Dyer, A.G., Dorin, A. (2021a). Towards computer vision and deep learning facilitated pollination monitoring for agriculture. *Proceedings of the ieee/cvf conference on computer vision and pattern*

*recognition* (pp. 2921–2930).

Ratnayake, M.N., Dyer, A.G., Dorin, A. (2021b). Tracking individual honeybees among wildflower clusters with computer vision-facilitated pollinator monitoring. *Plos one*, *16*(2), e0239504,

Reece, S.J., Radloff, F.G., Leslie, A.J., Amin, R., Tambling, C.J. (2021). A camera trap appraisal of species richness and community composition of medium and large mammals in a miombo woodland reserve. *African Journal of Ecology*, *59*(4), 898–911,

Schindler, F., & Steinhage, V. (2021). Identification of animals and recognition of their actions in wildlife videos using deep learning techniques. *Ecological Informatics*, *61*, 101215,

Sheard, J.K., Adriaens, T., Bowler, D.E., Büermann, A., Callaghan, C.T., Camprasse, E.C., ... others (2024). Emerging technologies in citizen science and potential for insect monitoring. *Philosophical Transactions of the Royal Society B*, *379*(1904), 20230106,

Simons, D.J., & Chabris, C.F. (1999). Gorillas in our midst: Sustained inattentional blindness for dynamic events. *perception*, *28*(9), 1059–1074,

Sittinger, M., Uhler, J., Pink, M., Herz, A. (2023). Insect detect: An open-source diy camera trap for automated insect monitoring. *bioRxiv*, 2023–12,

Sittinger, M., Uhler, J., Pink, M., Herz, A. (2024). Insect detect: An open-source diy camera trap for automated insect monitoring. *Plos one*, *19*(4), e0295474,

Sullivan, G.J., Ohm, J.-R., Han, W.-J., Wiegand, T. (2012). Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, *22*(12), 1649–1668,

Swinnen, K.R., Reijniers, J., Breno, M., Leirs, H. (2014). A novel method to reduce time investment when processing videos from camera trap studies. *PloS one*, *9*(6), e98881,

Tan, M., Chao, W., Cheng, J.-K., Zhou, M., Ma, Y., Jiang, X., ... Feng, L. (2022). Animal detection and classification from camera trap images using different mainstream object detection architectures. *Animals*, *12*(15), 1976,

Ungureanu, V.-I., Negirla, P., Korodi, A. (2024). Image-compression techniques: Classical and "region-of-interest-based" approaches presented in recent papers. *Sensors*, *24*(3), 791,

van der Voort, G.E., Gilmore, S.R., Gorrell, J.C., Janes, J.K. (2022). Continuous video capture, and pollinia tracking, in platanthera (orchidaceae) reveal new insect visitors and potential pollinators. *PeerJ*, *10*, e13191,

van Klink, R., August, T., Bas, Y., Bodesheim, P., Bonn, A., Fossøy, F., ... others (2022). Emerging technologies revolutionise insect ecology and monitoring. *Trends in ecology & evolution*, ,

Van Klink, R., Sheard, J.K., Høye, T.T., Roslin, T., Do Nascimento, L.A., Bauer, S. (2024). *Towards a toolkit for global insect biodiversity monitoring* (Vol. 379) (No. 1904). The Royal Society.

Wiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A. (2003). Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, *13*(7), 560–576,

Wittmann, K., Gamal Ibrahim, M., Straw, A.D., Klein, A.-M., Staab, M. (2024). Monitoring fast-moving animals—building a customized camera system and evaluation toolset. *Methods in Ecology and Evolution*, *15*(5), 836–842,

Wong, W.-M., & Kachel, S. (2024). Camera trapping—advancing the technology. *Snow leopards* (pp. 415–428). Elsevier.

Zett, T., Stratford, K.J., Weise, F.J. (2022). Inter-observer variance and agreement of wildlife information extracted from camera trap images. *Biodiversity and Conservation*, *31*(12), 3019–3037,

Zualkernan, I., Dhou, S., Judas, J., Sajun, A.R., Gomez, B.R., Hussain, L.A. (2022). An iot system using deep learning to classify camera trap images on the edge. *Computers*, *11*(1), 13,