Online-Journals.org

iJIM International Journal of Interactive Mobile Technologies

iJIM | eISSN: 1865-7923 | Vol. 19 No. 17 (2025) | OPEN ACCESS

https://doi.org/10.3991/ijim.v19i17.56777

PAPER

A Hybrid Approach for Moving Object Detection and Tracking in Event-Based Cameras

Ahmed S. Ghorab[1], **Raed S. Rasheed**[2] (✉), **Hanan Abu-Mariah**[3], **Wesam M. Ashour**[2]

[1]University College of Applied Sciences (UCAS), Gaza, Palestine

[2]Islamic University of Gaza, Gaza, Palestine

[3]Palestine Ahliya University, Bethlehem, Palestine

rrasheed@iugaza.edu.ps

ABSTRACT

Event cameras, also called dynamic vision sensor-based cameras, capture visual information differently than frame-based cameras. These asynchronous event streams record brightness changes with great temporal resolution and low latency, making them perfect for difficult applications. Data preparation, noise removal, and object tracking are issues when using event cameras in computer vision. This work offers a hybrid clustering-tracking method to accurately locate and track moving objects in event camera data. This study introduces a hybrid technique for accurate moving object detection and tracking in event camera data. Our method uses the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) density-based algorithm to eliminate noise and cluster and track in two steps. The clustering process uses standard methods led by centroids from prior frames for accuracy. The tracking method predicts cluster positions in later frames using speed and direction information when clusters overlap. Our hybrid technique achieves 95.35% accuracy against ground truth labels, promising major improvements in event-based camera data-based computer vision and robotics applications.

KEYWORDS

event-based cameras, object detection, clustering, tracking, noise removal, density-based spatial clustering of applications with noise (DBSCAN), computer vision, robotics

1 INTRODUCTION

Event cameras, also known as dynamic vision sensor-based cameras (DVS), are a type of novel bio-inspired sensor that works differently from traditional frame cameras [1]. Rather than recording an image intensity at a slow synchronous fixed rate like in active pixel sensors (APS), DVS asynchronously detects changes in brightness on a per-pixel basis and outputs a stream of events that encode the time, location, and polarity of these changes (i.e., whether the pixel became brighter or darker). Those events have the latency of microseconds, which lets you reconstruct a scene in real time with high temporal resolution and low latency. This makes event cameras

Ghorab, A. S., Rasheed, R. S., Abu-Mariah, H., Ashour, W. M. (2025). A Hybrid Approach for Moving Object Detection and Tracking in Event-Based Cameras. *International Journal of Interactive Mobile Technologies (iJIM)*, 19(17), pp. 60–80. https://doi.org/10.3991/ijim.v19i17.56777

Article submitted 2025-05-24. Revision uploaded 2025-07-09. Final acceptance 2025-07-18.

perfect for situations that are hard for traditional cameras to handle [2], [3]. One of the advantages of DVSs is their high dynamic range, which means they can capture both bright and dark regions of a scene simultaneously without losing detail. In addition to that, event cameras consume less power than traditional cameras because they only output events when there is a change in brightness, rather than continuously capturing frames. Moreover, traditional cameras suffer from image blur because they capture a series of frames at a fixed rate, and the objects in the scene may move during the time it takes to capture each frame. As a result, the final image may contain blurred or distorted objects, especially if the objects are moving quickly or if the camera is not held steady. This is known as "motion blur," and it can reduce the clarity and detail of the image. Robotics and computer vision applications are two examples of event camera applications [1]–[3]. Recent studies have also explored mobile front-ends for real-time sensing and analytics [4]–[5].

When dealing with event-based cameras, traditional computer vision algorithms face a number of problems, as event cameras do not provide the same level of information as conventional frame-based cameras. These algorithms require considerable event data preprocessing, and their performance is dependent on carefully specified parameters. Furthermore, event-based cameras lack essential visual features such as texture and color, making moving object recognition more difficult. Furthermore, the asynchronous activation of event-based sensors generates data streams with irregular space-time coordinates, which standard algorithms may struggle to analyze successfully [6], [7]. Recent work such as FlexEvent has shown the importance of frequency-adaptive frameworks for event data, achieving high accuracy across 20–180 Hz [8].

The sparse and asynchronous nature of the event data is incompatible with conventional computer vision algorithms. Therefore, in order to exploit existing computer vision algorithms, an easy way is to collect events into frames or volumes with fixed or variable lengths [2]. Utilizing such algorithms undermines some of the benefits of event-based vision sensors, specifically their power efficiency and high temporal resolution, which are basic and fundamental properties of the neuromorphic computing paradigm. Other methods involve transforming the event stream into geometric data structures, such as 3D point clouds [2].

The novel neuromorphic event cameras' multi-cluster tracking method was developed by Aladem et al. [9]. The Gaussian fluid tracker underpins these multi-object trackers. Higher-level geometric mimicry may be monitored forever. Additionally, detectors may be taught to execute geometric impersonation disclosures. Blobs are monitored continually. Therefore, they may be able to conduct visual odometer because the data union issue is readily solved. Ad-hoc and heuristic ways to build groupings of the same item are still useful and might pass. With proper knowledge about the objective application and event camera architecture, such methods may be creative and controlled to provide predictable results. This may provide a foundation for event-based multi-object trackers.

The study presented in this study has yielded several noteworthy contributions, which can be succinctly summarized as follows:

- This paper introduces a two-step strategy that combines clustering and tracking to precisely recognize and track mobile objects. The hybrid method predicts trajectories using centroid identification, convex hull creation, and previous direction data.
- Using the DBSCAN density-based method in data preprocessing. Eliminating redundant information from event-based camera data improves data quality and reliability.

- This study proposes a framework for optimizing DBSCAN algorithm parameters like epsilon (ε) and (MinPts). This improves noise removal and clustering accuracy.
- Centroid information from prior frames improves conventional clustering approaches. Integration of this technique improves moving object detection stability and precision.
- Using velocity and trajectory information from prior frames to predict moving object trajectories, even when different clusters overlap.
- Comparing the proposed approach to ground truth labels to prove its efficacy. The suggested method detects and tracks moving objects in event-based camera data more accurately.

The remainder of the paper is structured as follows: Section 2 contains a literature review of relevant works. Section 3 describes the research's historical context. Section 4 presents the methodology. Section 5 contains the experimental outcomes and analysis. Finally, Section 6 concludes with a discussion of conclusions and future work.

2 BACKGROUND

2.1 DBSCAN clustering algorithm

DBSCAN is an acronym that stands for Density-Based Spatial Clustering of Applications with Noise. It was proposed in 1996 by Ester et al. DBSCAN is designed to cluster data of arbitrary shapes in the presence of noise in spatial and non-spatial high-dimensional databases [10].

For each item of a cluster, the neighborhood of a specific radius (epsilon—ε) must contain at least a minimum number of objects (MinPts), meaning that the neighborhood's cardinality must exceed some threshold [11].

The DBSCAN algorithm classifies the data points as core, border, and noise. Core points are those with a minimum number of adjacent data points within a given radius (ε). Border points are those that have fewer neighbors than the minimal number but are nonetheless inside the radius provided for a core point (MinPts). Noise points are ones that do not belong to any cluster due to a lack of sufficient neighbors within the radius given [11].

2.2 k-means clustering algorithm

The k-means clustering method is a well-known unsupervised machine learning approach used to divide a dataset into k groups based on their similarity [12]. The procedure assigns each data point repeatedly to the closest centroid and then recalculates the centroid of each cluster until convergence is reached. At convergence, each data point is labeled with the index of its corresponding cluster.

However, the technique has drawbacks, such as difficulties connected with the random initialization of the centroids, which lead to unexpected convergence. In addition, such a clustering technique necessitates that the number of clusters be specified in advance, which accounts for the varying cluster forms and outlier effects [12].

To overcome the constraints of the k-means clustering technique, researchers have offered a variety of alternatives.

2.3 Minibatch k-means clustering algorithm

Minibatch k-means optimization is a variant of the traditional batch approach utilized in the k-means clustering algorithm. It cuts processing costs by orders of magnitude while producing solutions that are much superior to online stochastic gradient descent. Instead of utilizing the complete dataset to update the cluster centers, just a randomly selected portion (or minibatch) is utilized in minibatch k-means. This makes real-time clustering feasible for user-facing applications and enables the production of near-optimal clusters [13].

2.4 Agglomerative and BIRCH clustering algorithms

Hierarchical clustering is a technique for cluster analysis that seeks to establish a hierarchy of groups. It is a tree-based collection of simple (flat) clustering algorithms. These approaches generate clusters by recursively splitting items from the top down or the bottom up. In contrast to conventional clustering approaches, hierarchical clustering builds a tiered hierarchy of clusters rather than allocating each data point to a single cluster. This permits more detailed data analysis and interpretation, as well as the identification of subclusters inside larger clusters [14], [15].

Agglomerative hierarchical clustering is a bottom-up clustering method in which each data point begins in its own cluster, and pairs of clusters are merged until all data points belong to a single cluster. Following are the stages required to conduct agglomerative hierarchical clustering [15]:

- Assign each data point to its own cluster to get started.
- Using a distance measure such as the Euclidean distance, calculate the distance between each pair of clusters.
- The two nearest clusters should be merged into a single cluster.
- Recalculate the distances between this new cluster and every other cluster.
- Repeat steps 3 and 4 until every data point belongs to the same cluster.

Various linking criteria, including single linkage, full linkage, and average linkage, can be utilized to calculate the distance between two clusters. These criteria compute the distance between two clusters differently dependent on the distances of their individual data points. The choice of linking criterion can have a substantial effect on the clustering solution that is generated [15].

The Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) clustering method is classified as an agglomerative hierarchical clustering technique [16]. The algorithm is specifically intended to effectively process extensive datasets via the initial creation of a hierarchical arrangement of subclusters, followed by repeated merging to generate bigger clusters.

The BIRCH algorithm produces a hierarchical structure known as the Clustering Feature Tree (CF Tree) using a bottom-up approach. During each iteration, the algorithm combines smaller subclusters into bigger ones, ultimately leading to the formation of a hierarchical clustering structure. The technique retains statistical measures such as the centroid and diameter for each subcluster, facilitating the effective merging of clusters according to certain distance and density requirements [17].

2.5 CURE clustering algorithm

Clustering Using Representatives (CURE) is a technique that finds a ground between centroid-based and all-point approaches. It uses points for each cluster, which are selected from well-distributed cluster points and then adjusted towards the center of the cluster by a certain proportion. This allows CURE to effectively handle clusters with shapes and varying sizes. Additionally, CURE combines sampling. Partitioning techniques to efficiently handle large databases, and it incorporates outlier management algorithms to effectively filter outliers from the dataset [18].

CURE can be applied to any type of data that can be represented in a vector space. However, it's worth noting that CURE is particularly beneficial for datasets with spherical clusters and changing cluster sizes [18].

2.6 Gaussian mixture model clustering algorithm

The Gaussian Mixture Model (GMM) uses the Expectation-Maximization technique to create ellipsoidal clusters unsupervised. Probability density estimates determine these clusters' membership. Each cluster is modeled using Gaussian distribution. GMMs can quantify fitness based on cluster number better than k-means. This offers GMMs an edge over k-means. This gain comes from considering both the mean and the covariance, unlike the k-means method, which only considers the mean [19].

3 RELATED WORK

Few efforts have been made recently to expand the problem of detecting moving objects in neuromorphic vision [20]. This section provides a brief overview of the techniques that are currently available for detecting moving objects in traditional frame-based cameras and examines how these techniques are being adapted for use in neuromorphic vision. The use of deep learning is one of these approaches, as in [21], [22], [23], such that they used multi-layered convolutional neural networks (CNN) for detecting moving objects. But there are some problems with using deep learning techniques, such as the complexity of the models and the need for a large amount of labeled data to prevent overfitting, which is rarely available in the current time. Dedicated hardware, like GPUs, is also required for training and prediction.

Many classical approaches that are proposed for moving object detection are based on event-based clustering and tracking. A. Mondal et al. [7] suggested a method employing k-means clustering to detect moving objects in event-based cameras. The authors compare their suggested method to cutting-edge algorithms and provide encouraging findings. Additionally, asynchronous multi-object trackers such as AEMOT leverage blob detection and track validation to robustly handle occlusions in event streams [24].

The same authors suggested in [6] a new approach for detecting moving objects utilizing event-based cameras and Graph Spectral Clustering (GSC). The suggested method samples the event space-time volume based on the timestamps of grayscale images and builds a similarity graph utilizing k-NN. The authors then do eigen decomposition on the Graph Laplacian and take the first k eigenvectors to obtain clusters of moving objects. The ideal value of k is identified by silhouette analysis. Experimental results demonstrate that the suggested method

outperforms existing methods in terms of precision and efficiency. However, these methods have a number of drawbacks, particularly when we attempt to extend them to event-based vision and when we use different datasets. Recent research has demonstrated deep learning on mobile platforms for tasks ranging from parking-space classification [4] to adaptive push-notification design [5] and real-time social media analytics [25], underscoring the viability of lightweight, on-device processing pipelines.

G. Chen [26] proposed a novel technique for the detection and tracking of multiple vehicles using neuromorphic vision. Their system's performance is tested using data collected by a neuromorphic vision sensor positioned on a highway bridge. They conducted a preliminary investigation on multivehicle tracking by clustering, utilizing three traditional clustering methods (e.g., GMM [27], MeanShift [28], and DBSCAN [10]) and four tracking methods. Utilizing the low latency and sparse event stream, they were able to quickly integrate an online tracking-by-clustering system with a high frame rate, which greatly exceeds the real-time capabilities of typical frame-based cameras.

However, the use of GMMs in clustering can lead to limitations in the accuracy of the results due to their sensitivity to noise and their inherited assumption of independence among data points, which neglects the existence of similarity relations. Density-based methods such as DBSCAN are also not ideal for clustering sparse event-based data because the data points vary in density. Additionally, MeanShift approaches require the optimization of numerous parameters, posing a challenge in determining the optimal value and consequently increasing the complexity of the clustering process.

Long-term tracking is a revolutionary object tracking technique developed by Bharath et al. [29] primarily for event cameras. This tracking framework is the first of its type for event cameras, employing a novel object representation and online learning. It effectively re-tracks the item whenever it reappears inside the camera's range of vision. A major innovation is the use of an event-based regional sliding window technique, which handles complex scenarios with cluttered and dynamic backdrops successfully.

Vasco et al. [30] presented an event-based separate move detector using the event camera. The main idea of the paper is to propose a method for segmenting the motion of independently moving objects using event-driven cameras. The paper addresses the challenge of confounding background clutter events due to robot ego-motion when cameras are mounted on a moving robot. The proposed method detects and tracks corners in the event stream and learns the statistics of their motion as a function of the robot's joint velocities. By comparing the predicted corner velocities from ego-motion with the measured corner velocities, independently moving objects can be identified.

4 METHODOLOGY

4.1 Used dataset: DVSMOTION20 dataset [31]

The *DVSMOTION20* is a four-dimensional dataset; it is prepared to examine the advanced event-based optical flow algorithms. The data were gathered using the *IniVation DAViS346* camera, which has a 346×260 spatial resolution. The dataset consists of camera motion data (stationary scene and moving camera) and object motion data (stationary camera and moving objects) [32].

The camera motion data belongs to four actual indoor ranges (namely, checkerboard, classroom, conference room, and conference room translation) with complete ground real complete movement from the inertial measurement unit (*IMU*). The move of the camera in this class was detained by a gimbal, and the *IMU* was standardized before each collection [32].

The object motion data consists of two real sequences (called hands and cars) containing different object moves. This class does not have actual ground movement since the object movement cannot be concluded from the *IMU*. [32] The object motion data of size (*169 MB*) is readable in *MATLAB* format [33]. The raw data file contains two object data motion sequences called hands and cars sequences; each sequence is stored in a separate *MATLAB* file, which is constructed from the following "*davis*" structure [32].

The present study employed the Hands Sequence as the chosen methodology to investigate the proposed model. The Hands Sequence is composed of *6002301* rows, each containing four columns (variables): (*x*) and (*y*) representing the event coordinates, (*p*) representing polarity, and (*t*) representing the timestamp. There are a total of *190* grayscale images corresponding to *190* distinct timestamps. The quantity of occurrences varies across each timestamp image. The occurrence signifies alterations in the logarithmic intensity.

4.2 Event data preparation

To begin using event-based cameras, the raw event data captured by the camera sensor must be converted into a software-compatible format. Event data typically consists of the x and y coordinates of the pixel where the event occurred as well as the type of event, which might be positive or negative. A rise in light intensity is categorized as a positive event, whereas a drop in light intensity is categorized as a negative event [3].

In order to transform event data from an event camera into a format that can be processed in software, the accumulation time (t_a) parameter must be specified. This parameter specifies the amount of time in microseconds over which events will be gathered. This process of aggregation involves collecting event data over a specified period of time and arranging it into synthetic frames that are shown at regular intervals. Once the frames have been created, they are prepared for the detection phase of the analysis procedure [3].

Whether the event data is received in real-time from an event-based camera or a pre-recorded dataset, it always takes the form of a time-ordered sequence of events. The developer is responsible for structuring and arranging these events in a manner that facilitates their intended analysis or application. This may involve translating raw event data into a more understandable format, such as aggregating events into frames or volumes, or transforming the event stream into geometric data structures [3]. As the raw data is sparse and asynchronous, it might be challenging to deal with using typical computer vision techniques without proper structure and processing of the event data.

In order to utilize algorithms that are built to process frames, the event data must be converted into synthetic frames. The size of these synthetic frames is normally determined by the horizontal and vertical resolution parameters of the camera used to capture the events, as in Figure 1. By constructing these synthetic frames, event-based data may be processed using algorithms that were originally designed for traditional video data [3]. See Figure 2 for an example of a synthesized frame in gray level.

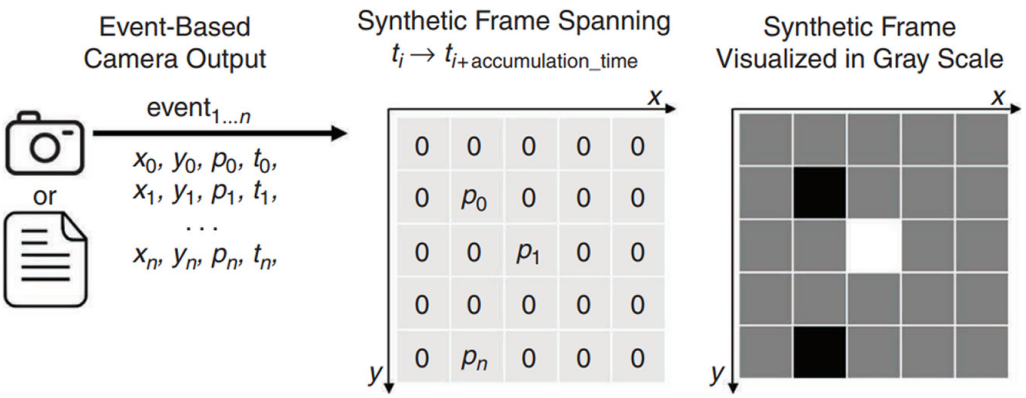

Fig. 1. A synthetic frame is created by filling an empty matrix with the outputs from an event-based camera

Note: When the frame is viewed, pixels that correspond to positive-polarity events appear as white, and those that correspond to negative-polarity events appear as black [3].

Fig. 2. An example of a synthesized grayscale frame generated from event camera data for object detection

4.3 The proposed algorithm

This paper introduces an innovative hybrid method for detecting and tracking moving objects in event-based cameras. The approach consists of two steps:

In the first step we begin by applying a clustering algorithm to the initial synthetic frame, as in Figure 3, in order to identify clusters, followed by convex hull construction and centroid determination for each cluster. Next, we reapply the algorithm on the following synthetic frame and calculate the speed and direction of the moving clusters by comparing their centroid locations between two successive frames.

The second step starts when overlapping occurs between clusters in one of the synthetic frames, as in Figure 4. At this point we pause the clustering algorithm and initiate a tracking algorithm to continue tracking the previously detected moving clusters from the previous frame. The tracking algorithm utilizes information about speed and direction to predict where the clusters will be located in the next synthetic frame. Once the overlapping period ends, we resume using the clustering algorithm on the next synthetic frames, repeating this process for detecting and tracking moving clusters.

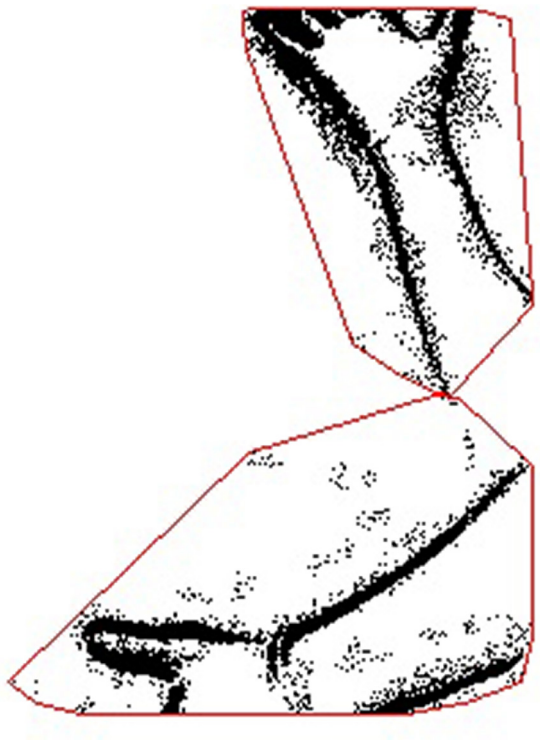

Fig. 3. The result of clustering the initial synthetic frame
Note: The red line represents the constructed convex hull around the clusters.

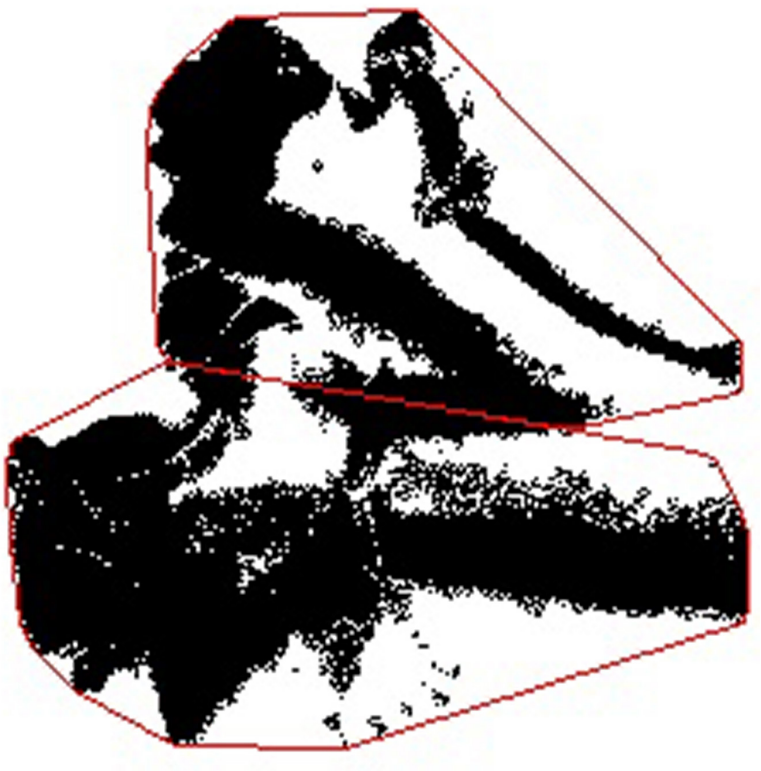

Fig. 4. Overlapped clusters are detected, and the second step is initiated (i.e., the clustering algorithm is paused and the tracking algorithm is started to continue tracking the previously detected moving clusters from the previous frame)

In order to apply the proposed hybrid approach, firstly, the event data need to be preprocessed to remove noisy events to ensure the quality and reliability. Next, a suitable clustering algorithm, as described in the background section of this paper, is applied to identify distinct moving object clusters in the data. During the clustering process, the hybrid approach continuously monitors for overlapping clusters in each frame. Once an overlap is detected, the tracking algorithm is triggered, and the clusters from the previous frame are tracked forward in time in the subsequent frame. The tracking algorithm utilizes the speed and direction information from the previous frame to predict the clusters' next positions in the subsequent frame.

The hybrid clustering algorithm first reads the event data, as shown in algorithm-1 and the flowchart in Figure 5. The epsilon (ε) value for each event data frame is calculated using the elbow approach. The epsilon (ε) value is a parameter that governs the inter-point distance inside a cluster. Subsequently, the data is subjected to noise removal by the use of the *DBSCAN* algorithm. The density-based clustering *DBSCAN* approach finds data points with high local neighborhood connection.

After noise removal, the system creates a synthetic frame. To improve clustering accuracy, a synthetic frame is used. The dataset is then clustered using traditional methods. Regular clustering uses a distance measure to organize data points into clusters.

If this is the initial frame, the approach merely uses standard clustering. When the current frame is not the beginning frame, the centroids from the previous frame guide the clustering process. This intervention stabilizes clustering outcomes.

Finally, the method checks cluster overlap. When overlap occurs, the algorithm watches the cluster object. This approach helps find things that switch clusters.

The utilization of a hybrid clustering methodology exhibits significant effectiveness in the clustering of event data. Recent mixture-of-experts heat-conduction detectors have demonstrated similar robustness to motion blur and low-light conditions [34]. The method described in this study combines the advantages of *DBSCAN* with standard clustering methods, leading to clustering results that exhibit high precision and robustness. The hybrid clustering approach has numerous advantages:

- The system exhibits the ability to efficiently administer and alleviate noise that is inherent in the data.
- The algorithm possesses the capacity to identify and categorize clusters of diverse shapes and dimensions.
- The utilization of this technology enables the surveillance and tracking of cluster entities.

In general, the hybrid clustering strategy provides a robust technique for the clustering of event data. This option is highly suitable for applications that prioritize precision and consistency. The implementation is fully coded in Python and located in the GitHub repository[1].

To evaluate the performance and accuracy of the hybrid approach, a comparison is made between the resultant clusters and the ground truth real labels present in the original dataset. This evaluation step allows for a thorough assessment of the approach's effectiveness in accurately detecting and tracking moving objects in event-based camera data.

[1] Hybrid-Approach-for-Moving-Object-Detection, Last Access 22.06.2025 [Online]. Available: https://github.com/raedrasheed/Hybrid-Approach-for-Moving-Object-Detection.git.

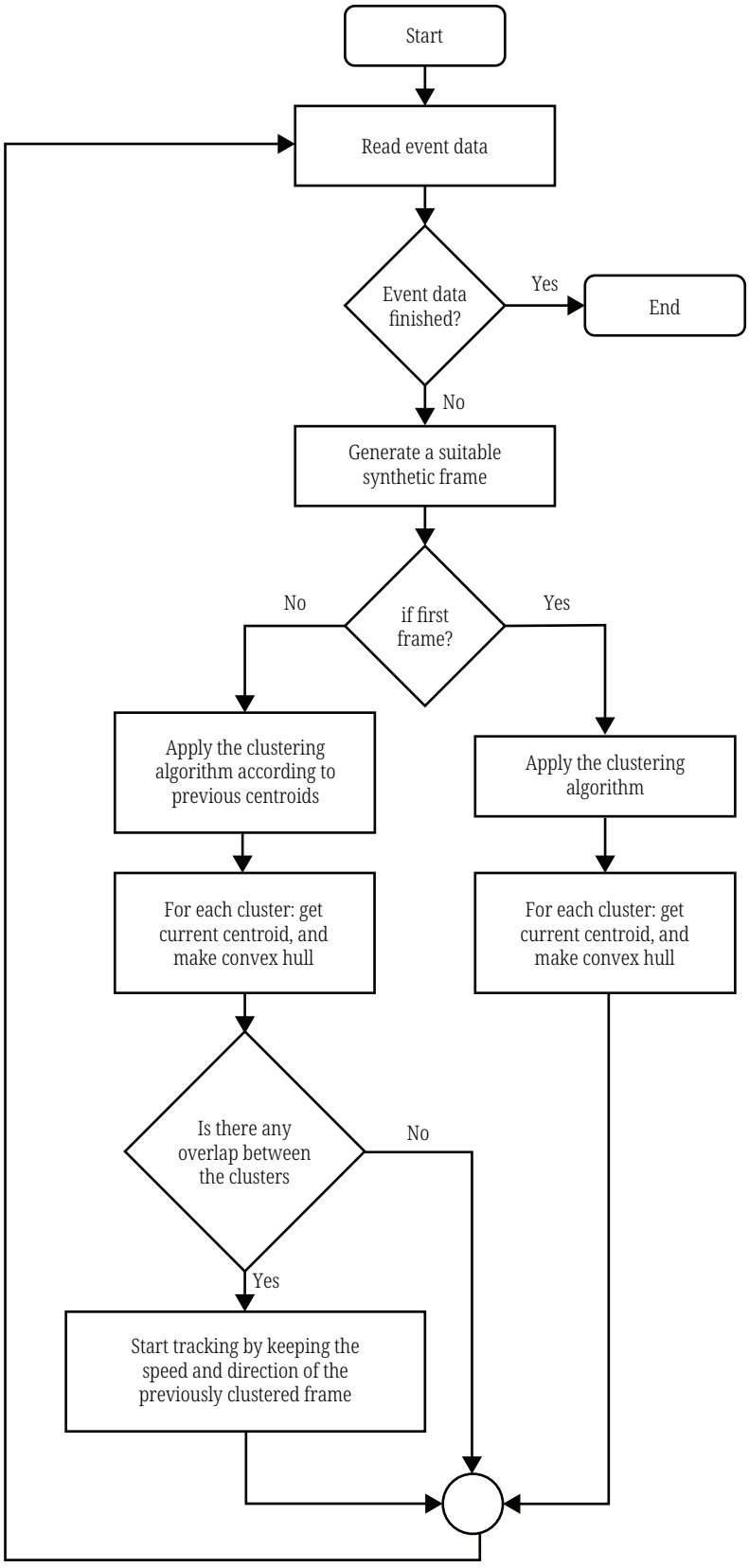

Fig. 5. The flowchart of our proposed hybrid algorithm, detailing event noise removal, clustering, centroid tracking, and adaptive decision stages

The proposed hybrid approach integrates noise removal, clustering, and tracking algorithms to efficiently and accurately detect and track moving objects in event-based camera data, offering promising potential for a wide range of computer vision and robotics applications. Here are the steps in details:

Algorithm 1: Hybrid Clustering Approach

```
Input: Event Data
Output: Clusters
Read Event Data
while event_data NOT finished do
    Read event_data_frame
    Calculate ε using Elbow method
    Remove Noise using DBSCAN algorithm
    Generate synthetic_frame
    if first_frame then
        Apply regular clustering
        foreach cluster do
            Get current_centroid
            Set convex_hull
        end foreach
    else
        Apply regular clustering according to previous_centroids
        foreach cluster do
            Get current_centroid
            Set convex_hull
        end foreach
        if there is any overlap between the clusters then
            Tracking cluster_object
        end if
    end if
end while
```

Algorithm-1's computational complexity is dominated by the DBSCAN clustering on N events per synthetic frame, which—when accelerated via a KD-tree spatial index—runs in $O(N \log N)$ time (degrading to $O(N^2)$ without indexing). Subsequent pipeline stages—frame synthesis, cluster centroid computation, convex-hull approximation, and overlap-based tracking—each incur at most $O(N + K \log K)$ overhead per frame, where K is the number of clusters. Memory usage scales linearly with N for storing raw events, spatial indices, and label arrays, with an additional $O(K)$ buffer required for centroid histories and hull data.

Data preprocessing step. Event camera data, acquired from dynamic vision sensor-based cameras, presents unique characteristics that distinguish it from traditional frame-based camera data. Unlike conventional cameras that capture images at a fixed rate, event cameras detect changes in brightness on a per-pixel basis and output a stream of events encoding the time, location, and polarity of these changes.

However, the asynchronous nature of event cameras and the sparsity of the event data pose challenges for subsequent processing steps. The data acquired by event cameras may contain noise, which can manifest as false positive or false negative events. This noise can arise due to various factors, such as sensor imperfections, electrical noise, or environmental disturbances.

Therefore, a crucial preprocessing step in working with event camera data involves noise removal to enhance the quality and reliability of the data. By effectively filtering out noise, the subsequent analysis and interpretation of event camera data can be performed more accurately, leading to improved performance in various computer vision tasks such as object detection, tracking, and scene reconstruction.

Hence, the approach employed for noise elimination involves integrating it within the *DBSCAN* technique. *DBSCAN* categorizes each data point into three distinct types: core points, border points, and noise points [10]. In *DBSCAN*, a core point is defined as a point that has a minimum number of neighboring points (*MinPts*) within a specific radius (ε), making it directly density reachable. Border points, on the other hand, are points that are densely reachable but not directly reachable. Finally, noise points refer to the remaining points that are neither density reachable nor assigned to any cluster. By incorporating these definitions, *DBSCAN* provides an effective means to identify and isolate noise from the event camera data, facilitating more accurate and reliable clustering results.

Prior to executing the noise removal algorithm on the datasets, it is essential to fine-tune the algorithm's parameters, namely (ε) and (*MinPts*), which will be discussed in the next step.

DBSCAN parameters tuning. The epsilon (ε) is the most important parameter in *DBSCAN*, because the whole process after that depends on the (ε) value. For this mission, find the *K*-nearest neighbors of each event. We sorted *k* nearest neighbor distances and plotted distance fluctuation. After the elbow point appeared, the optimal *y-axis* distance was (ε) [35]. For instance, if we take the third frame, the value of the (ε) equals to 5.22, as in Figure 6.

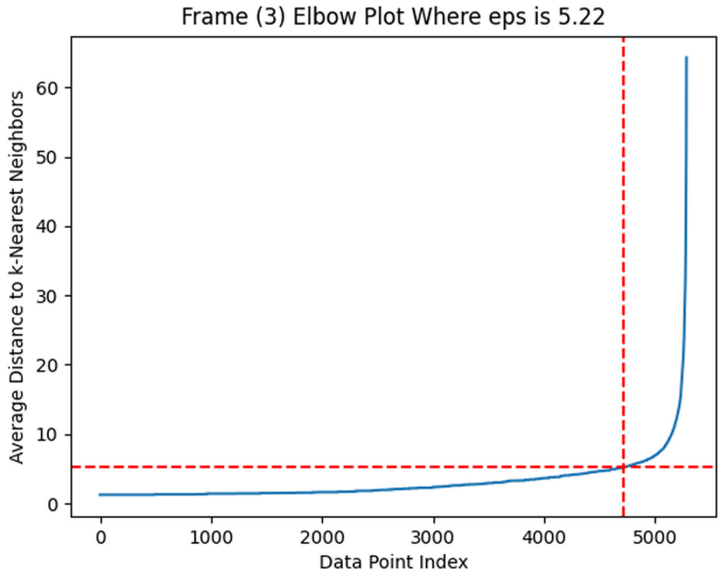

Fig. 6. Epsilon (ε) value for the frame #3; the (ε) value at this frame is equal to 5.22

MinPts depends on dataset dimensions (no. of dimensions). If your data has more than two dimensions, choose $MinPts = 2*dim$, where *dim* is the number of dimensions of your data set [36]. The minimum number of samples (*MinPts*) assigned generally depends on the number of dimensions (usually equal to twice the number of the dimensions).

This formula assigned the minimum sample number parameter. After testing, we empirically selected (*MinPts*) as the best. Then, (*MinPts*) is the value where the noise reduction algorithm detects noise with less inaccuracy.

The impact of the noise removal process is depicted in Figure 7. It is evident that the frame contains a substantial amount of noise, and removing such noise improves tracking accuracy on the performance of both the clustering algorithm and the tracking process.

Fig. 7. Illustrates how the noise removal pre-processing step significantly enhances clustering accuracy by eliminating irrelevant events and preserving object motion information

Applying clustering algorithms. This section outlines clustering approaches for moving object detection. The suggested dataset was clustered using three techniques in this step. The types are: *k-means*, *minibatch k-means*, *GMM,* and *BIRCH* clustering, which are discussed in the background section in this paper.

Numerous studies have introduced methodologies utilizing the silhouette coefficient for the purpose of estimating the optimal number of clusters in numerical data clustering [37]. The technique of silhouette analysis is employed to assess the efficacy and reliability of clustering outcomes. The functioning of the system involves the computation of a score for each individual data point, which quantifies the degree of similarity between the point and its own cluster (referred to as intra-cluster similarity) in relation to the nearest neighboring cluster (known as inter-cluster similarity).

A silhouette coefficient value approaching *(1)* signifies that the given data point has strong clustering characteristics, characterized by a notable degree of similarity within clusters and a considerable dissimilarity between clusters. A silhouette coefficient in near proximity to *(–1)* signifies that the data point exhibits inadequate clustering, characterized by a diminished level of similarity within clusters and an elevated level of similarity between clusters.

A silhouette coefficient in proximity to *zero* suggests that the given data point resides at the boundary separating two distinct clusters. In order to ascertain the most suitable number of clusters, it is possible to generate a plot depicting the silhouette coefficient for every value of *k*. The best number of clusters can be determined by identifying the value of *k* that yields the highest average silhouette coefficient [38].

Clusters tracking step. Upon detecting overlapping clusters, the tracking algorithm is activated. The tracking algorithm utilizes the speed and direction information from the previous frame's clusters to predict the next positions of the corresponding objects in the subsequent frame. To identify each cluster, we used the centroid in order to determine the location of the moving cluster. In addition to that, we established a bounding shape created by the convex hull methodology, which helps to enclose the cluster with a convex polygon, enabling us to better visualize and understand the spatial extent of the moving objects. The combination of centroid-based localization and the convex hull shape allows us to accurately identify and analyze the clusters in the event-based data.

Figures 8 and 9 illustrate the difference between the identified clusters obtained without applying our hybrid methodology and those achieved by employing the proposed hybrid approach. It is worth noting that the clusters detected without

our hybrid methodology, as in Figure 8, resulted in a less accurate representation of the actual moving objects. In contrast, the clusters obtained through the hybrid approach, as in Figure 9, demonstrate a more cohesive and concentrated arrangement, closely aligning with the true positions and shapes of the moving objects in the scene. This difference highlights the effectiveness of our hybrid methodology in improving the accuracy and reliability of cluster detection, further validating its utility for robust moving object detection and tracking in event-based camera data.

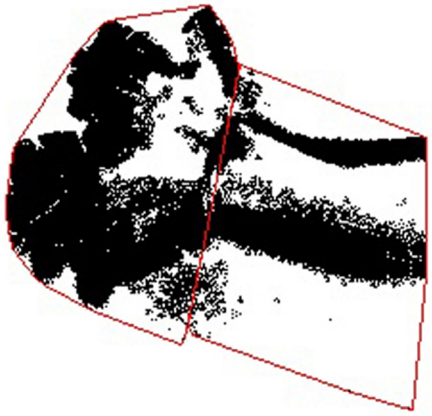

Fig. 8. Clusters detected using only the original clustering algorithms, (i.e., without using our hybrid methodology)

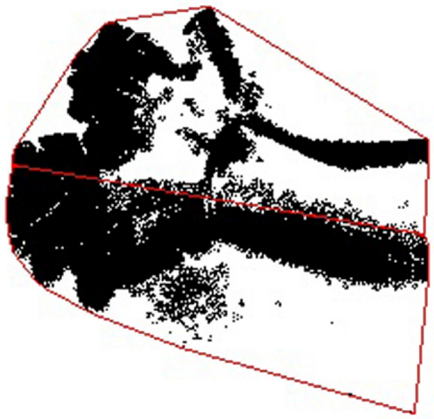

Fig. 9. Clusters detected using our hybrid methodology

4.4 Performance evaluation metrics

The two systems' performance of the proposed framework was evaluated using *F1-score, precision, and recall* measures; see equations from (1) to (3) [39]

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{1}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{2}$$

$$F1\_score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3}$$

4.5 Generating ground truth (real) labels

The process of generating *real* labels for the synthesized frames involves several, first of all, the input event data is loaded from a file in sequence, and then it is converted into grayscale synthesized frames, as in Figure 2. Then the user is provided with an interactive *GUI* interface to draw the freehand masks corresponding to the clusters of interest (e.g., two hands in this use case), as shown in Figure 10.

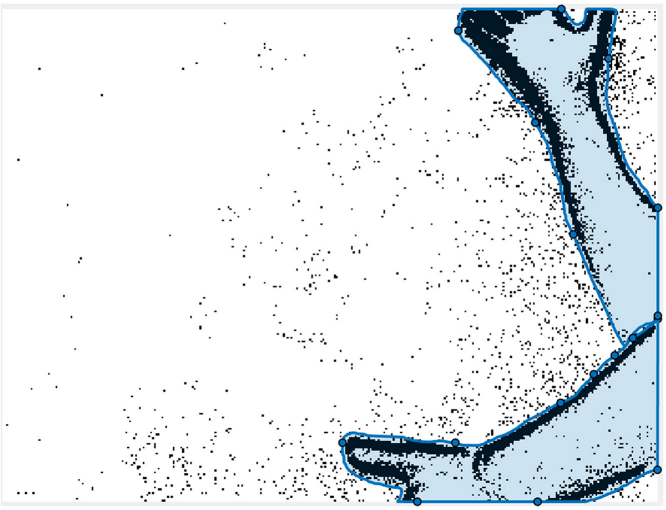

Fig. 10. Example of drawing freehand masks to generate the real labels. (e.g. Frame # 7)

Subsequently, binary masks are generated from the drawn regions, and a combined mask is formed by combining the two binary masks, as shown in Figure 11.

Fig. 11. The generated masks for the existing clusters. (e.g. Frame # 7)

Finally, each event within the selected frame's timestamp range is assigned a label based on its location within these binary masks; all other events are assigned a neutral label that represents the noise. For instance, the labeled events for frame *(#7)* of the Hands Sequence in this use case are visualized on a scatter plot, with different colors representing the three assigned labels: *blue* for *Cluster 1*, *red* for *Cluster 2*, and *black* for others in Figure 12.

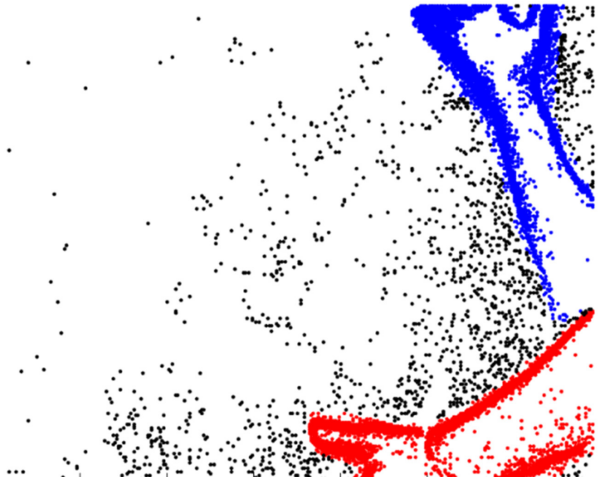

Fig. 12. The generated real labels used to evaluate the performance of the proposed methodology (e.g. Frame # 7)

The described methodology encapsulates the steps involved in generating real labels for the synthesized frames, enabling subsequent evaluation and comparison of the clustering and tracking algorithms' performance.

5 RESULTS AND ANALYSIS

Table 1 provides a complete comparison of five different clustering algorithms, namely *k-means, BIRCH, minibatch k-means, CURE, and GMM*, across four separate scenarios. The algorithms are evaluated using performance criteria such as *accuracy, precision, recall, and F1-score.*

Table 1. Experiment results

		k-Means	BIRCH	Minibatch	k-Means	BIRCH
Regular Approach with Noise	Accuracy	81.69%	81.01%	81.69%	78.71%	80.55%
	Precision	79.04%	78.95%	79.10%	73.68%	78.29%
	Recall	82.55%	82.05%	82.61%	78.35%	81.66%
	F1-score	78.88%	78.28%	78.90%	73.57%	78.01%
Regular Approach without Noise	Accuracy	91.04%	90.22%	91.08%	86.65%	89.85%
	Precision	90.67%	90.28%	90.77%	82.28%	89.80%
	Recall	90.83%	89.91%	90.94%	84.47%	89.68%
	F1-score	90.42%	89.48%	90.49%	81.82%	89.38%
Our Hybrid Methodology with Noise	Accuracy	88.15%	88.31%	88.16%	85.30%	86.66%
	Precision	85.88%	85.60%	85.88%	80.30%	84.50%
	Recall	87.70%	87.96%	87.70%	83.59%	86.45%
	F1-score	85.63%	85.87%	85.63%	80.38%	84.16%
Our Hybrid Methodology without Noise	Accuracy	95.20%	95.35%	95.20%	90.90%	93.45%
	Precision	95.66%	95.16%	95.66%	86.94%	93.69%
	Recall	95.09%	95.45%	95.09%	88.87%	93.35%
	F1-score	94.71%	95.01%	94.71%	86.14%	92.63%

The occurrence of noise inside data is a prevalent obstacle in the process of clustering. Based on the findings presented, it is clear that the presence of noise consistently diminishes the performance of all algorithms. In the context of the "Regular Approach with Noise" scenario, the accuracy rates for several clustering algorithms are as follows: *k-means* achieves an accuracy of 81.69%, *BIRCH* achieves 81.01%, *minibatch k-means* achieves 81.69%, *CURE* achieves 78.71%, and *GMM* achieves 80.55%. In marked contrast, the removal of noise in the "Regular Approach without Noise" scenario leads to a significant improvement in accuracy for all algorithms: *k-means* achieves an accuracy of 91.04%, *BIRCH* achieves 90.22%, *minibatch k-means* achieves 91.08%, *CURE* achieves 86.65%, and *GMM* achieves 89.85%.

One notable finding derived from Table I is the notable superiority in performance exhibited by the alternative method, referred to as "Our Hybrid Methodology," when compared to the conventional approach. The clustering findings of our algorithm demonstrate improvement regardless of the presence or absence of noise. In the scenario titled "Our Hybrid Methodology with Noise," the accuracy metrics demonstrate notable improvements. Specifically, the *k-means* algorithm achieves an accuracy of 88.15%, the *BIRCH* algorithm achieves 88.31%, *the minibatch k-means* algorithm achieves 88.16%, the *CURE* algorithm achieves 85.30%, and the *GMM* achieves 86.66%. The aforementioned superiority is further accentuated in a noise-free setting, where the scenario labeled as "Our Hybrid Methodology without Noise" exhibits the highest levels of accuracy. Specifically, the *k-means* algorithm achieves an accuracy of 95.20%, the *BIRCH* algorithm achieves an accuracy of 95.35%, the *minibatch k-means* algorithm achieves an accuracy of 95.20%, the *CURE* algorithm achieves an accuracy of 90.90%, and *GMM* achieves an accuracy of 93.45%.

Several algorithms, such as *k-means* and *minibatch k-means*, demonstrate a notable level of consistency in their performance, regardless of the methodology employed or the presence of noise. The remarkable aspect of their ability to withstand noise is deserving of special attention. According to the conventional methodology, the disparity in accuracy between the presence of noise and its absence in the *k-means* algorithm is just below *10%.* The aforementioned remark is also applicable to *minibatch k-means*.

In finishing this study, it is imperative to acknowledge the significance of contextualizing these findings within the specific characteristics of the dataset and any domain-specific prerequisites. Although "Our Hybrid Methodology" shows promise, particularly in dealing with datasets that contain noise, it is crucial to have a comprehensive understanding of the dataset, its characteristics, and the available resources in order to make an informed decision about selecting a clustering algorithm.

6 CONCLUSIONS

In conclusion, this study offers a novel hybrid technique for event-based camera moving object detection and tracking. This finding aligns with recent automotive-vision studies showing the benefits of event cameras in ADAS contexts [40]. The suggested approach improves dynamic object detection accuracy and reliability by exploiting early grouping and subsequent tracking. Centroid determination, convex hull creation, and historical data make the hybrid technique suitable for cluster identification and trajectory prediction. The incorporation of the DBSCAN density-based algorithm improves data quality by removing noise from event-based camera data. The technique is optimized for noise removal and clustering through systematic parameter tuning using elbow-curve analysis and Gaussian-based distance estimation. The hybrid technique improves object detection stability by having high clustering precision. The tracking method predicts trajectory

accurately even in overlapping clusters using past data. Evaluation against ground truth labels shows these contributions' impact. In numerous cases, the hybrid technique outperforms classic clustering algorithms with the maximum accuracy. The "Our Hybrid Methodology without Noise" scenario shows accuracy rates of up to 95.35%, demonstrating its potential to develop computer vision. By seamlessly integrating noise removal, clustering, and tracking, this study provides a robust framework with promising implications for computer vision and robotics applications, improving event-based camera data object detection and tracking.

7 REFERENCES

[1] G. Gallego *et al.*, "Event-based vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 154–180, 2020. https://doi.org/10.1109/TPAMI.2020.3008413

[2] H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza, "Events-to-video: Bringing modern computer vision to event cameras," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, 2019, pp. 3852–3861. https://doi.org/10.1109/CVPR.2019.00398

[3] C. Iaboni, H. Patel, D. Lobo, J.-w. Choi and P. Abichandan, "Where are they going? Clustering event camera data to detect and track moving objects," *Computer*, vol. 55, pp. 90–94, 2022. https://doi.org/10.1109/MC.2021.3136451

[4] H. Abu-Alsaad, "CNN-based smart parking system," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 17, no. 11, pp. 155–170, 2023. https://doi.org/10.3991/ijim.v17i11.37033

[5] A. Wohllebe, M. Rolf Adler, and S. Podruzsik, "Influence of design elements of mobile push notifications on mobile app user interactions," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 15, no. 15, pp. 35–46, 2021. https://doi.org/10.3991/ijim.v15i15.23897

[6] A. Mondal, J. Giraldo, T. Bouwmans, and A. Chowdhury, "Moving object detection for event-based vision using graph spectral clustering," in *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2021, pp. 876–884. https://doi.org/10.1109/ICCVW54120.2021.00103

[7] A. Mondal and M. Das, "Moving object detection for event-based vision using k-means clustering," in *2021 IEEE 8th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, 2021, pp. 1–6. https://doi.org/10.1109/UPCON52273.2021.9667636

[8] D. Lu, L. Kong, G. H. Lee, C. S. Chane, and W. T. Ooi, "FlexEvent: Towards flexible event-frame object detection at varying operational frequencies," *arXiv preprint arXiv:2412.06708*, 2024. https://doi.org/10.48550/arXiv.2412.06708

[9] M. Aladem and S. A. Rawashdeh, "A multi-cluster tracking algorithm with an event camera," in *IEEE National Aerospace and Electronics Conference (NAECON)*, 2019, pp. 391–397. https://doi.org/10.1109/NAECON46414.2019.9058204

[10] K. Khan, S. Rehman, K. Aziz, S. Fong, S. Sarasvady, and A. Vishwa, "DBSCAN: Past, present and future," in *the Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014)*, 2014, pp. 232–238. https://doi.org/10.1109/ICADIWT.2014.6814687

[11] T. Verma and D. Gaur, "A survey on study of enhanced DBSCAN algorithm," *International Journal of Engineering Research & Technology*, vol. 2, no. 11, pp. 1483–1488, 2013.

[12] M. Ahmed, R. Seraj, and S. M. Shamsul Islam, "The k-means algorithm: A comprehensive survey and performance evaluation," *Electronics*, vol. 9, no. 8, p. 1295, 2020. https://doi.org/10.3390/electronics9081295

[13] D. Sculley, "Web-scale K-means clustering," in *Proceedings of the 19th International Conference on World Wide Web*, 2010, pp. 1177–1178. https://doi.org/10.1145/1772690.1772862

[14] F. Nielsen, *Introduction to HPC with MPI for Data Science*. Cham: Springer, 2016. https://doi.org/10.1007/978-3-319-21903-5

[15] P. Shetty and S. Singh, "Hierarchical clustering: A survey," *International Journal of Applied Research*, vol. 7, no. 4, pp. 178–181, 2021. https://doi.org/10.22271/allresearch.2021.v7.i4c.8484

[16] S. K. Mann and S. Chawla, "A proposed hybrid clustering algorithm using K-means and BIRCH for cluster-based cab recommender system (CBCRS)," *International Journal of Information Technology*, vol. 15, pp. 219–227, 2023. https://doi.org/10.1007/s41870-022-01113-6

[17] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: A new data clustering algorithm and its applications," *Data Mining and Knowledge Discovery*, pp. 141–182, 1997. https://link.springer.com/article/10.1023/A:1009783824328

[18] S. Guha, R. Rastogi, and K. Shim, "CURE: An efficient clustering algorithm for large databases," *ACM SIGMOD Record*, vol. 27, no. 2, pp. 73–84, 1998. https://doi.org/10.1145/276305.276312

[19] E. Patel and D. S. Kushwaha, "Clustering cloud workloads: K-Means vs Gaussian Mixture Model," *Procedia Computer Science*, vol. 171, pp. 158–167, 2020. https://doi.org/10.1016/j.procs.2020.04.017

[20] Y. Gao *et al.*, "Action recognition and benchmark using event cameras," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 12, pp. 14081–14097, 2023. https://doi.org/10.1109/TPAMI.2023.3300741

[21] H. Zhu, X. Yan, H. Tang, Y. Chang, B. Li, and X. Yuan, "Moving object detection with deep CNNs," *IEEE Access*, vol. 8, pp. 29729–29741, 2020. https://doi.org/10.1109/ACCESS.2020.2972562

[22] M. A. Pérez-Cutiño, A. G. Eguíluz, J. R. Martínez-de Dios, and A. Ollero, "Event-based human intrusion detection in UAS using deep learning," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2021, pp. 91–100. https://doi.org/10.1109/ICUAS51884.2021.9476677

[23] H.-Y. Huang, C.-Y. Lin, W.-Y. Lin, C.-C. Lee, and C.-Y. Chang, "Deep learning based moving object detection for video surveillance," in *2019 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, 2019, pp. 1–2. https://doi.org/10.1109/ICCE-TW46550.2019.8991778

[24] A. Apps, Z. Wang, V. Perejogin, T. L. Molloy, and R. Mahony, "Asynchronous multi-object tracking with an event camera," *arXiv preprint arXiv:2505.08126*, 2025. https://doi.org/10.48550/arXiv.2505.08126

[25] S. Rautela, "Social media for new product launch: A study of social media platforms across the RACE planning framework," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 15, no. 5, pp. 187–204, 2021. https://doi.org/10.3991/ijim.v15i05.18147

[26] G. Chen *et al.*, "Neuromorphic vision based multivehicle detection and tracking for intelligent transportation system," *Journal of Advanced Transportation*, p. 4815383, 2018. https://doi.org/10.1155/2018/4815383

[27] D. Reynolds, "Gaussian mixture models," in *Encyclopedia of Biometrics*, S. Z. Li and A. Jain, Eds., 2009, pp. 659–663. https://doi.org/10.1007/978-0-387-73003-5_196

[28] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002. https://doi.org/10.1109/34.1000236

[29] B. Ramesh, S. Zhang, Z. W. Lee, Z. Gao, G. Orchard, and C. Xiang, "Long-term object tracking with a moving event camera," in *British Machine Vision Conference*, 2018. http://bmvc2018.org/contents/papers/0814.pdf

[30] V. Vasco, A. Glover, E. Mueggler, D. Scaramuzza, L. Natale, and C. Bartolozzi, "Independent motion detection with event-driven cameras," in *2017 18th International Conference on Advanced Robotics (ICAR),* 2017, pp. 530–536. https://doi.org/10.1109/ICAR.2017.8023661

[31] M. Almatrafi, R. Baldwin, K. Aizawa, and K. Hirakawa, "Datasets," 2023. [Online]. Available: https://sites.google.com/a/udayton.edu/issl/software/dataset

[32] M. Almatrafi, K. Aizawa, and K. Hirakawa, "Distance surface for event-based optical flow," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 7, pp. 1547–1556, 2020. https://doi.org/10.1109/TPAMI.2020.2986748

[33] H. Abu-Mariah and W. Ashour, "Moving object detection based on clustering and event-based camera," in *2023 8th International Engineering Conference on Renewable Energy & Sustainability (ieCRES),* 2023, pp. 1–5. https://doi.org/10.1109/ieCRES57315.2023.10209469

[34] X. Wang *et al.*, "Object detection using event camera: A MoE heat conduction based detector and a new benchmark dataset," *arXiv preprint arXiv:2412.06647*, 2025. https://doi.org/10.48550/arXiv.2412.06647

[35] D. T. Lan and S. Yoon, "Trajectory clustering-based anomaly detection in indoor human movement," *Sensors*, vol. 23, no. 6, p. 3318, 2023. https://doi.org/10.3390/s23063318

[36] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, pp. 1–21, 2017. https://doi.org/10.1145/3068335

[37] D.-T. Dinh, T. Fujinami, and V.-N. Huynh, "Estimating the optimal number of clusters in categorical data clustering by silhouette coefficient," in *Knowledge and Systems Sciences: 20th International Symposium, KSS 2019,* in Communications in Computer and Information Science, J. Chen, V. Huynh, G. N. Nguyen, and X. Tang, Eds., Springer, Singapore, vol. 1103, 2019, pp. 1–17. https://doi.org/10.1007/978-981-15-1209-4_1

[38] P. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987. https://doi.org/10.1016/0377-0427(87)90125-7

[39] M. Hossin and S. Md Nasir, "A review on evaluation metrics for data classification evaluations," *International Journal of Data Mining & Knowledge Management Process (IJDKP)*, vol. 5, no. 2, pp. 1–11, 2015. https://doi.org/10.5121/ijdkp.2015.5201

[40] D. Gehrig and D. Scaramuzza, "Low-latency automotive vision with event cameras," *Nature*, vol. 629, pp. 1034–1040, 2024. https://doi.org/10.1038/s41586-024-07409-w

8 AUTHORS

Ahmed S. Ghorab is with the University College of Applied Sciences (UCAS), Gaza, Palestine.

Raed S. Rasheed is with the Islamic University of Gaza, P.O. Box 108, Gaza, Palestine (E-mail: rrasheed@iugaza.edu.ps).

Hanan Abu-Mariah is with the Palestine Ahliya University, P.O. Box 1041, Bethlehem, Palestine.

Wesam M. Ashour is with the Islamic University of Gaza, P.O. Box 108, Gaza, Palestine.