

Abstract—Contrary to other standard cameras, event cameras interpret the world in an entirely different manner; as a collection of asynchronous events. As event camera's unique data output, many event feature detection and tracking algorithms have shown significant progress by making detours to frame-based data representations. This paper questions the need to do so and proposes a novel event data-friendly method that achieve simultaneous feature detection and tracking, called event Clustering-based Detection and Tracking (eCDT). Our method employs a novel clustering method, named as k-NN Classifier-based Spatial Clustering and Applications with Noise (KCSCAN), to cluster adjacent polarity events to retrieve event trajectories. With the aid of a Head and Tail Descriptor Matching process, event clusters that reappear in a different polarity are continually tracked, elongating the feature tracks. Thanks to our clustering approach in spatio-temporal space, our method automatically solves feature detection and feature tracking simultaneously. Also, eCDT can extract feature tracks at any frequency with an adjustable time window, which does not corrupt the high temporal resolution of the original event data. Our method achieves 30% better feature tracking ages compared with the state-of-the-art approach while also having a low error approximately equal to it.

I. INTRODUCTION

An event camera is a neuromorphic sensor that mimics the functions of an eye [1]; it detects the intensity changes in its pixels that are output as signals called events. These events are either on or off events caused by a positive or negative change of intensity, respectively. Also, since these events are created asynchronously and independently per pixel, event cameras are granted the following three merits [1]: a) robustness against highly dynamic situations, b) high temporal resolution, and c) low power requirements. Noticing the potentials of event cameras, the computer vision community has been paying increased attention towards this sensor.

Contrary to most camera variants, such as thermal cameras [2], which are frame-based, event cameras are pixel-based. To be more specific, whereas frame-based cameras output frames at synchronous time intervals, event cameras output each pixel data asynchronously. Due to this difference, conventional computer vision algorithms are not compatible with event cameras. As a result, a common method of coping with event data is to create event frames via accumulating pixel data within a time window. Another widely utilized data representation is the surface of active events (SAE), where frames are created with pixel intensity values representing how recent an event occurred in that pixel. Overall, while the frame-based representations of event data enables the adoption of vast precedent computer vision techniques, the data representation is suboptimal [1], [3]. In other words, these algorithms are not fully exploiting the advantages of event cameras.

In our paper, we diverge from frame-based representations and explore the capabilities of clustering as a feature detection and tracking algorithm.

Our key contributions are as follows:

- We formulate a novel clustering method called k-NN Classifier-based Spatial Clustering Application with Noise (KCSCAN) that clusters adjacent equi-polarity events but also segregates the clusters by opposing polarity events, thus resulting in a spatio-temporal clustering of features.
- By finding continual cluster tracks using cluster descriptors, event Clustering-based Feature Detection and Tracking (eCDT) shows about 30 percent longer tracks compared to the state-of-the-art algorithm (HASTE).
- Additionally, since KCSCAN clusters in a spatio-temporal space, it simultaneously performs feature detection and tracking, automatically detecting features and tracking them at the same time.

We believe this approach could be a stepping stone to more variants of algorithms that is more fitted to event-based data.

II. RELATED WORKS

A. Clustering as a Means of Object Detection and Tracking

Object detection and tracking for event cameras were explored early on. In 2006, [4] proposed a vehicle tracking system on highways by clustering events that were triggered by passing cars. They took advantage of the fact that a stationary event camera setup only triggers events that correspond to moving objects. The stationary event camera setup has been extensively used in further studies by [5], [6]. In contrast to the aforementioned works, [7], [8] developed methods that even worked when the event camera was not stationary. Nonetheless, clustering was a key component of detecting and tracking the objects.

B. Predefined Shape Detection and Tracking

[9] identifies the state of grippers by tracking the positions of pre-defined left and right gripper locations in the image by...
using an Iterative Closest Point (ICP) algorithm. [10] tracks the location of a ball for iCub, an event-driven humanoid robot. With the shape of the object being circular, the authors extends the Hough-based circle detection algorithm for event cameras by using optical flow. In [11], line features are detected by getting intuitions from the Line Segment Detection (LSD) method. It calculates the orientation of the spatial derivative for each pixel and clusters events of similar orientation to form lines.

C. Corner Detection

Feature detection and tracking of specific points of interest, such as corners, has shown progress in recent years with the adoption of a different type of data representation, which is called the Surface of Active Events (SAE). It imitates a frame-like representation where its pixel ‘intensity’ values are higher with more recent events triggered at that pixel. Leveraging SAE’s frame-like nature, many corner detection algorithms have been developed [12]–[15] by taking intuitions from common corner detectors [16], [17].

[14] was the first to introduce a method of corner detection for event cameras. Their method, called eHarris, is an adaptation of the original HARRIS algorithm [16]. Following up, [13] was the first to adapt the original Features from Accelerated Segment Test (FAST) algorithm [17] and called their method eFAST. Both eHarris and eFAST detected the corner points by first converting the events into frame-based images by the SAE representation. Based on the two approaches suggested above, an abundant variation of approaches were suggested such as [12], [15]. As a variation of eFAST [13], [12] can detect corners with obtuse angles. Also, this variant utilized a modified version of the SAE. [15] is a further development of [12], but also accompanying the eHarris algorithm to process corner events. In contrary to these classical methods, [18] proposes a method that trains a random forest decision tree that assess if an event is a corner point or not. This method utilizes another variant of SAE called Speed-invariant Surface of Active Events. As summarized above, corner point detection is mostly done in the SAE representation.

D. Feature Point Tracking

A variety of feature tracking algorithms were also published [19]–[22]. First of all, the eKLT [22] tracks features using both frames and events. Harris corners are detected from the frames and a patch around the corner is used to create a template, which is tracked with events. [20] approaches feature tracking by calculating the corner point descriptor and finding corner points in adjacent regions of the most similar descriptor. [21] is a method that tracks features using template batch matching. The templates are generated by the motion compensated event frame along a Bézier curve. This creates a clearer image to track. At each timestamp, the events are stacked along the curve and creates a Motion Compensated Event Frame. Similarly, multi-Hypothesis Asynchronous Speeded-up Tracking of Events (HASTE) [19] tracks its points by using template matching. In this work, multiple hypotheses are given according to the direction of movement: translation and rotation. With the combination of movements, the likelihood of each hypothesis is calculated and whenever a hypothesis overtakes the previously prevalent hypothesis, the unit movement corresponding to the new hypothesis is considered to be detected. With each new hypothesis becoming the null hypothesis, the template is updated and the tracking algorithm continues this process.

III. eCDT: event Clustering-based Detection and Tracking

A. Intuitive Explanation of the Approach

As mentioned previously, an event is triggered at a certain pixel \((x,y)\) and time \((t)\) if there is a (logarithmic) brightness change, due to moving edges, i.e., intensity gradients, in the image plane, that reaches a predefined threshold value \(C\). With each pixel independently outputting events, the event data is a collection of these events: \(E_N = \{e_1, e_2, ..., e_N\}\), where \(e_i = (x_i, y_i, t_i, p_i)\) is the \(i\)-th event representation that consists of the \(x, y\) pixel coordinates, the time \(t\), the polarity \(p\), and \(N\) is the total number of events.

In its raw form, i.e. the spatio-temporal space, as shown in Fig. 2(a), event data are unintelligible, as stated by [23]. Therefore, motion compensation studies [23], [24] have shown that warping the events according to the trajectory of the edges leads to a crisp and clear revelation of the edge structures as shown in Fig. 2(c). Fig. 2(c) shows that by warping events according to their trajectory, events are clearly more visible by being aggregated with their own polarities. Therefore, motion compensation tries to maximize the image variance between the two polarities.
B. KCSCAN: a Novel Means of Detection and Tracking

We now describe the crux of our proposed method. Our method depends on the observation that spatio-temporal adjacency of equal polarity events can function as a measure to discern whether events were triggered by the same edge structure or not. Thanks to the high temporal resolution of event cameras, a nearest neighbor approach to track objects or detected event features (e.g., corners) has been already prevalently practiced in this field [3]–[8], [18].

Therefore, with the assumption that close equal polarity points are triggered from the same edge, we adopt Density-based Spatial Clustering of Applications with Noise (DBSCAN) [25] as a baseline method of clustering equal polarities. For completeness, we briefly describe DBSCAN. It is a non-parametric clustering algorithm that classifies points into a cluster if they are closely packed together. Clusters consist of points that are classified as core points when there are more than a predefined number of points $n$ within a predefined radius $r$. Neighboring points are also evaluated in an expansive manner, which therefore groups points of closely situated core points into a single cluster.

However, DBSCAN alone cannot discern equal polarity clusters that are close but separated by opposite polarities. Therefore, we implement a subtle idea that recognizes opposite polarity events as borders when performing event clustering. We demonstrate the idea with the aid of some illustrations in 2D space. As shown in Fig. 3(a), there are red points distributed from the bottom left corner to the upper right corner. Similarly, blue points are distributed from the bottom right corner to the upper left corner. Points of each color are positioned in a way that separates the differently colored points. By assessing if a point is neighboring any opposite polarity points, we can detect boundaries to a cluster. Therefore, with barriers separating the target polarity points, the 2D points should be classified as shown in Fig. 3(b). Also, points that do not belong to any cluster are considered noise.

To implement this idea, we use a k-NN classifier that discerns a point’s purity by votes. More formally, assume we identified the $k$-nearest neighbors $E_k$ of a query point $e_q = (x_q, y_q, t_q, p_q)$, where $E_k \subseteq E_N$. Then,

$$E_k = \{(x_1, y_1, t_1, p_1), (x_2, y_2, t_2, p_2), \ldots, (x_k, y_k, t_k, p_k)\}.$$  \hfill (1)

We can define the purity score $\Phi_q$ for a query point $e_q$ as

$$\Phi_q = \frac{1}{k} \sum_{i=1}^{k} \delta_{p_i = p_q};$$  \hfill (2)

where $\delta$ denotes the Kronecker Delta function. The purity score basically measures how pure, i.e., homogeneous, its surrounding is. With a minimum purity score predefined as $\Phi_{min}$, $e_q$ is a core point if $\Phi_q \geq \Phi_{min}$. So, referring back to Fig. 3(a), assuming a purity score $\Phi_{min} = 1$, we notice that point A’s 4-nearest neighbors (excluding itself) are of equal polarity, making A a core point. On the other hand, the 4-nearest neighbors of point B have different polarity points, making B just noise as shown in Fig. 3(b). To prevent

In addition to their contribution, in our work, we focus on another key observation: edge patterns of different polarities create a boundary separating clusters of events. In Fig. 2(b) and (d), we see clusters are visually separated by cluster of opposite polarities despite being close distanced. Therefore, neighboring opposite polarity events act as a cluster boundary that divides the spatio-temporal space. Referring to Fig. 2(a), we notice the edge structures are actually manifested as a space dividing plane, which is what separates one cluster from another. Hence if we could cluster neighboring events of equal polarity bounded by opposite polarity clusters, we could modestly cluster events from the same edge and subsequently retrieve the trajectory.
Fig. 4. (a) Raw events of 5 seconds are shown. Red indicates on events and blue off events. We can clearly notice a sudden change of direction in the middle from right to left. We also notice that the change of direction results in the change in polarity despite being from the same edge feature. (b) Clusters resulting from eCDT without the head and tail descriptor module show that the polarity inversion due to a sudden direction change results in discontinuous feature clustering. The pink cluster in the middle does not continue up, but starts as a new cluster in white. (c) eCDT clusters shows that longer tracking is possible. The head and tail descriptor match has lengthened the tracking for the pink cluster in the middle. A set of colors are randomly assigned to each cluster to differentiate between nearby clusters (best viewed in color).

situations where k-NN are absurdly far away due to the sparsity of event data, we bound our k-NN search to a radius \( r \). The algorithm that determines a core event is given in Algorithm 1. The iterative approach to expand the clusters with core points is adopted from DBSCAN, where neighboring core points have the same cluster index. This novel clustering method that uses a k-NN classifier for its core point evaluation is named as k-NN Classifier-based Spatial Clustering Applications with Noise (KCSCAN).

C. Head and Tail Descriptor Matching

As shown in Figs. 4(b) and 4(c), clustering equal polarities show weakness when there is a sudden change of direction that inverts the polarity of events created by the same edge. Therefore, we create a cluster descriptor that enables continuous tracking despite inverted polarities.

Since we already recognize equal polarity cluster features with our clustering mechanism, data association between two clusters is imperative. Again, exploiting the spatio-temporal neighboring characteristic of events, events from the same edge will have spawned from a spatial position it disappeared, except for cases when features are lost because they left the camera’s field of view. Thus, we only need to search the proximity of space where a cluster ended to recapture the lost features. The temporal search limit is the search time \( s \). Therefore, we formulate the problem as finding matching descriptors in a spatio-temporal space near where a cluster terminates.

In situations of sudden movement, the polarity inverts, whereas the shape created by the edge remains constant in a short time window. To elaborate, the small time interval where motion direction changes becomes a situation where no events are created and the position and visual shape of a feature practically remains intact. Therefore, the problem only requires matching the shape and position of a terminating cluster and newly created adjacent ones. Since the cluster’s temporal boundary conditions, i.e. ‘head’ or ‘tail’ of a temporal trail of events, are only required to match clusters, we create cluster head descriptors to find preceding clusters from the same edge and cluster tail descriptors to find future clusters, originating from the same edge.

We incorporate a mutual information-based descriptor matching method as shown in Fig. 5. Suppose we have two clusters, \( C_i \) and \( C_j \), from the same edge, with cluster \( C_i \) preceding cluster \( C_j \). Then, the tail descriptor of cluster \( C_i \) should match the head descriptor of cluster \( C_j \) to be a match. For a short time interval \( \Delta t \), the tail pixel locations of cluster \( C_i \) and head pixel locations of cluster \( C_j \) are collected as sets, \( TD_{C_i} \) and \( HD_{C_j} \), where \( TD \) and \( HD \) stand for tail descriptor and head descriptor, respectively. With \( C_i = \{ e_k \}_{k=0}^N \), the head descriptor \( (HD) \) is defined as

\[
HD_{C_j} = \{ (x_k, y_k) | t_0 < t_k < t_0 + \Delta t \}
\]

and the tail descriptor \( (TD) \) is defined as

\[
TD_{C_i} = \{ (x_k, y_k) | t_N - \Delta t < t_k < t_N \}.
\]

The mutual information, \( I(C_i, C_j) \), also called intersection over union (IoU), is defined as

\[
I(C_i, C_j) = \frac{TD_{C_i} \cap HD_{C_j}}{TD_{C_i} \cup HD_{C_j}}
\]

Given the fact that events from the same feature are created in proximity, the descriptor matching is only performed for close instances of the tail of a preceding cluster and the head of a succeeding cluster. The clusters of the largest mutual information are recognized as the same feature and grouped together for longer tracking. As shown in Fig. 4(c), the corresponding clusters are grouped together, enabling longer tracks.

D. Feature Track Extraction

The feature tracks are estimated by calculating the center of gravity of events in a temporal window. Let cluster \( C = \{ e_k \}_{k=0}^N \), where \( e_k \) is in chronologically ascending order. The feature tracks are extracted by calculating the moving average (MA) within a time window \( t_w \). The moving average of event \( e_k \), \( MA_{t_k} \) is calculated by first collecting the set of events within the time interval \( t_w \) at time \( t_k \) as:

\[
S_{t_k} = \{ e_i | t_k - 0.5t_w < t_i < t_k + 0.5t_w \} \in C.
\]
Fig. 5. The tail and head cluster descriptors ($TDC_i$ and $HDC_j$) are extracted from the points’ within a time window $\Delta t$ as shown in (a), (b), and (c). The green colored intersection over union (IOU) is calculated as shown in (d) to check if clusters $C_i$ and $C_j$ are a match. The best IOU score between clusters in proximity confirms a match between two clusters, given the score goes over a certain threshold IOU value.

Then,

$$MA_k = \frac{\sum_{i=0}^{M} \{(x_i, y_i) \mid (x_i, y_i) \in e_i \in S_k\}}{M},$$

(7)

where $M$ is the number of elements in $S_k$.

The window could progress with each event, letting the feature tracks be as fast as the events are detected, which is essentially extracting the feature tracks at very high frequencies.

IV. EVALUATION

A. Dataset and Parameter Settings

We evaluated our algorithm with the public event-based camera dataset [26]. This dataset was recorded with a Dynamic and Active-pixel Vision Sensor (DAVIS) [27], with a 240 x 180 resolution, 130 dB dynamic range, and 3 $\mu$s latency. The dataset also provides ground truth pose of the camera and intensity images like standard cameras. For this dataset, we test our algorithm against 3 different motions (6dof, rotation, translation) in 3 different scenes (shapes, poster, boxes).

The parameters for our method were chosen based on trial-and-error. The parameters are shown in Table I.

| Algorithm 1: eCDT Core Point Evaluation |
|----------------------------------------|
| **Input:** # of neighbors $k$, radius $r$, min purity $\Phi_{min}$ |
| **Output:** is_CorePoint, neighbors $N_k$ |
| 1 | **def** CorePoint ($e_q$): |
| 2 | $N_k \leftarrow \{\}$ |
| 3 | $\Phi \leftarrow 1$ |
| 4 | $E_k \leftarrow kNN\_within\_radius(k, r)$ |
| 5 | for $p_n \in E_k$ do |
| 6 | | if $p_n \neq p_q$ then |
| 7 | | | $\Phi \leftarrow \Phi - \frac{1}{2}$ |
| 8 | end |
| 9 | $N_k \leftarrow N_k + \{p_n\}$ |
| 10 | if $\Phi < \Phi_{min}$ then |
| 11 | | return false, $\{\}$ |
| 12 | end |
| 13 | end |
| 14 | return true, $N_k$ |

TABLE I. Parameter settings for eCDT

| Parameter | Value |
|-----------|-------|
| k-NN points | 30 |
| Minimum purity score ($\Phi_{min}$) | 0.90 |
| Minimum feature age | 0.01 |
| Time window ($t_w$) | 0.01 |
| Search time (s) | 0.2 |
| Threshold Intersection over Union (IoU) | 0.7 |

B. Comparison with State-of-the-Art Methods

We compare our method with a previously published open-source tracker: multi-Hypothesis Asynchronous Speed-up Tracking of Events (HASTE) [19]. While studies such as [20], [21] are also directly related to our work, public implementations are not provided. In contrast, [28] is publicly available, but as HASTE outperforms their study, we compare our results only to HASTE. We use the publicly available implementations provided by the authors which will serve as a baseline to compare our algorithm.

C. Evaluation Method

There are two main methods of evaluating the feature tracks: KLT and 3D reprojection. The KLT method was suggested by [22]. Since the Event Camera Dataset [26] provides greyscale images, some works [21], [22] utilize these images to compute the ground truth tracks of the feature detection and tracking algorithms. Initialized features are tracked in the images by the Lucas-Kanade tracking algorithm [29] with a window size of 35 which is sufficiently large enough to track features of even non-parametric shapes. Since we are using images as a metric for ground truth evaluation, motion blur due to fast movements becomes a threat to this evaluation method. Therefore, this method is not utilized for our evaluation. The alternative evaluation method adopted by [3], [18]–[20] evaluates feature tracks by triangulating the 3D point of a feature track with ground truth camera poses obtained by the OptiTrack system at 200Hz. Once the 3D point is estimated, it is reprojected to each ground truth camera pose to calculate the reprojection error between the estimated feature point and the tracked feature point. Since this evaluation metric does not rely on the greyscale image, it can even evaluate tracks with motion blur and therefore assess all tracks from the dataset. We report the root mean squared error (RMSE) of each feature track and its feature age - the time elapsed between a feature’s initialization and its disposal.

Given the ground truth camera poses at 200 Hz frequency, we evaluate the 3D reprojection error, which is calculated by triangulating the 3D position of a feature with its track. Let the tracks of feature $i$ be,

$$track_i = \{p_0, p_1, \ldots, p_N\},$$

(8)

where $p_i = (x_i, y_i, t_i)$ is the spatio-temporal position, and $N$ is the number of feature points in the feature tracks. With the corresponding ground truth poses to each timestamp in $track_i$ as

$$\chi = \{\chi_0, \chi_1, \ldots, \chi_N\},$$

(9)
where \( \chi = (x, y, z, qx, qy, qz, qw) \) is the translation and rotation information of the camera at the corresponding timestamp of track, triangulating the 3D position of the feature is the same as finding how consistent the feature is with respect to the motion of the camera pose. So finding the 3D reprojection root mean squared error can be evaluated as follows:

\[
E_{rmse} = \sqrt{\frac{\sum_{i=1}^{N} d(p_i, \hat{p}_i)^2}{N}},
\]

where \( d(a, b) \) denotes the Euclidean distance between points \( a \) and \( b \). \( \hat{p}_i \) is the projected 2D point of the estimated 3D point. Finding the 3D point \( \hat{P} = (x, y, z) \) is done by minimizing \( E_{rmse} \). Also, \( \hat{p} \) is the projection of a 3D point \( \hat{P} \) to a 2D camera view of pose \( R \) and \( T \).

Each dataset is evaluated with an error threshold of 3, 5, and 7 pixels. Additionally, since the proposed method is a complete visual front-end capable of providing features and their tracks, we provide HASTE with the same features detected by our algorithm for a fair comparison. eCDT method without the HT Matching module will be denoted as eCDT (w/o HT).

V. RESULTS AND DISCUSSION

A. Feature Age

![Boxplots of feature ages for boxes, poster, and shapes](image)

As shown in Fig. 6, the feature age is longer for eCDT. The detailed numerical results of the feature age are also presented in Table II. For almost all cases, eCDT outperformed HASTE in feature age. Since the feature age does not form a Gaussian distribution, we present both the mean and median feature age of each method on 3 scenes, 3 motions, and 3 thresholds, which is a total of 27 samples.

| Method          | Mean Med. | 7     | 5     | 3     |
|-----------------|----------|-------|-------|-------|
| HASTE [19]      | 0.816    | 0.265 | 0.867 | 0.295 |
| eCDT w/o HT     | 0.417    | 0.240 | 0.427 | 0.235 |
| eCDT (Ours)     | 1.224    | 0.550 | 1.309 | 0.585 |

TABLE II. Comparison with the state of the art method in terms of feature age. Reprojection errors above the threshold values (7, 5, and 3) are considered as outliers and removed. The more, the better (unit: s).

B. Reprojection Error

By averaging all scenes, motions, and thresholds, the mean reprojection error is lowest for HASTE with 1.83 px whereas that of eCDT is 2.17 px, which is 0.34 px difference as shown in Table VI. However, in Fig. 7, the results do not exhibit any consensus towards one’s superiority over another since
TABLE VI. The average median, mean, and standard deviation of the reprojection error (px) of each method on 3 scenes, 3 motions, and 3 thresholds, which is a total of 27 samples.

| Method         | median (px) | mean (px) | stdev (px) |
|----------------|-------------|-----------|------------|
| HASTE [19]     | 1.62        | 1.83      | 0.55       |
| eCDT w/o HT    | 2.07        | 2.21      | 0.57       |
| eCDT (ours)    | 2.02        | 2.17      | 0.52       |

Further analyzing the results, considering the fact that HASTE tracks a center point of an \( n \times n \) patch while eCDT tracks an edge feature, HASTE should be superior in that it is tracking a more strict feature. When the error threshold is set to 3 pixels, HASTE shows lower reprojection errors. As HASTE tracks point features, it can inherently achieve a lower reprojection error in the lower bound of the spectrum. HASTE also shows better performance when the scene is \( \text{shapes} \). eCDT for the \( \text{shapes} \) scene generates large edge features which are naturally prone to larger reprojection error since the final feature track is an average point of the large edge feature. Despite these disadvantageous aspects, interestingly, eCDT shows better performance for 4 out of 9 cases, which shows that eCDT tracks features accurately despite tracking edge features.

C. Quality of Feature Tracking

Fig. 8. Example tracking results of eCDT from \( \text{shapes} \), \( \text{poster} \), and \( \text{boxes} \) respectively. The purple-colored ground truth tracks are generated by KLT just for reference.

![Example tracking results](image)

Fig. 9. \( \text{boxes} \) scene feature tracks for (a) HASTE and (b) eCDT. The tracking results show that while HASTE is steady, eCDT has noisy tracks. The purple-colored ground truth tracks are generated by KLT just for reference.

![Feature tracks](image)

We performed a paired \( t \)-Test to evaluate the mean reprojection errors between different methods. The \( p \)-value of the paired \( t \)-Test is presented in Table VIII. While eCDT w/o HT and eCDT has a \( p \)-value of 0.77, signaling no significant difference between the two methods, HASTE has \( p \)-values of 0.01 and 0.02 compared to eCDT w/o HT and eCDT, respectively. While these \( p \)-values represent a statistically significant difference in reprojection error, they are not as significant as the \( p \)-values from the feature age as shown in Table V. While the reprojection error’s \( p \)-value is in the order of -2, the feature age’s \( p \)-value is in the order of -7, which is different by 5 orders.

Fig. 7. Boxplots of reprojection errors for \( \text{boxes} \), \( \text{poster} \), and \( \text{shapes} \). The smaller, the better.

![Boxplots of reprojection errors](image)

Fig. 8 presents the tracking result of eCDT on top of greyscale images to check whether the tracks are visually accurate according to the motion of the camera. Assessing Fig. 9, we notice that the tracks from eCDT are noisy compared to those from HASTE. Although the features tracked from eCDT are edge features, when the feature tracks...
are generated, the edge feature tracks are simplified as a single centroid point. Due to this aspect, eCDT’s feature point are more susceptible to noise and, therefore, eCDT’s tracks are somewhat noisy.

VI. CONCLUSION AND FUTURE WORKS

In this study, we introduce an entirely novel method of feature detection and tracking for event cameras, which builds upon a key observation that spatio-temporal events are separated by opposing polarity events. By implementing our idea via a novel clustering method, we obtained a simultaneous detection and tracking algorithm. eCDT showed 30% longer feature tracks on average. Additionally, eCDT showed low reprojection errors on par with the state-of-the-art algorithm, despite tracking edge features. Lastly, the proposed method shows that there lay potential in utilizing the raw events as they are, i.e. without leveraging frame-like representations. We hope this approach functions as a stepping stone to new possible algorithms that utilize raw events.

In the future, we plan to implement this algorithm to be run in real-time for further application. Furthermore, since the motion trajectory of each cluster can be individually extracted, this method has potential for motion segmentation, where dynamic objects are present in the view and segmenting them is the task.

ACKNOWLEDGMENT

This research was supported by BK21 FOUR and StradVision. We appreciate all the supports of StradVision members who provided insight and expertise. The contents are solely the responsibility of the authors.

REFERENCES

[1] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K.Daniilidis, and D. Scaramuzza, “Event-based vision: A survey,” IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 44, no. 1, pp. 154–180, 2020.

[2] A. Lee, Y. Cho, S. Yoon, Y. Shin, and A. Kim, “ViViD: Vision for visibility dataset,” in IEEE International Conference on Robotics and Automation Workshop (ICRA-W), 2019.

[3] O. Yilmaz, C. Simon-Chane, and A. Histace, “Evaluation of event-based corner detectors,” Journal of Imaging, vol. 7, no. 2, p. 25, 2021.

[4] M. Litzenberger, C. Posch, D. Bauer, A. N. Belbachir, P. Schon, B. Kohn, and H. Garn, “Embedded vision system for real-time object tracking using an asynchronous transient vision sensor,” in IEEE Digital Signal Processing Workshop & IEEE Signal Processing Education Workshop, 2006, pp. 173–178.

[5] T. Bolten, R. Pohle-Frohlich, and K. D. Tönnies, “Application of hierarchical clustering for object tracking with a dynamic vision sensor,” in Proc. International Conference on Computational Science (ICCS), 2019, pp. 164–176.

[6] S. Schraml and A. N. Belbachir, “A spatio-temporal clustering method using real-time motion analysis on event-based 3D vision,” in Proc. IEEE/CFV Conference on Computer Vision and Pattern Recognition Workshop (CVPR-W), 2010, pp. 57–63.

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

[8] J. Rodríguez-Gómez, A. G. Eguluz, J. Martínez-de Dios, and A. Ollero, “Asynchronous event-based clustering and tracking for intrusion monitoring in UAS,” in Proc. IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 8518–8524.

[9] Z. Ni, A. Bolopion, J. Agnus, R. Benosman, and S. Regnier, “Asynchronous event-based visual shape tracking for stable haptic feedback in microrobotics,” IEEE Transactions on Robotics, vol. 28, no. 5, pp. 1081–1089, 2012.

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

[11] C. Brändli, J. Strubel, S. Keller, D. Scaramuzza, and T. Delbrück, “ELiSeD — An event-based line segment detector,” in Proc. International Conference on Event-based Control, Communication, and Signal Processing (EBCDSP), 2016, pp. 1–7.

[12] I. Alzugaray and M. Chi, “Asynchronous corner detection and tracking for event cameras in real time,” IEEE Robotics and Automation Letters, vol. 3, no. 4, pp. 3177–3184, 2018.

[13] E. Mueggler, C. Bartolozzi, and D. Scaramuzza, “Fast event-based corner detection,” in Proc. British Machine Vision Conference (BMVC), 2017, pp. 33.1–33.11.

[14] V. Vasco, A. Glover, and C. Bartolozzi, “Fast event-based Harris corner detection exploring the advantages of event-driven cameras,” in Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 4144–4149.

[15] R. Li, D. Shi, Y. Zhang, K. Li, and R. Li, “FA-Harris: A fast and asynchronous corner detector for event cameras,” in Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 6223–6229.

[16] C. G. Harris, M. Stephens, et al., “A combined corner and edge detector,” in Proc. Alvey Vision Conference, 1988, pp. 147–151.

[17] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in Proc. European Conference on Computer Vision (ECCV), 2006, pp. 430–443.

[18] J. Manderscheid, A. Sironi, N. Bourdis, D. Migliore, and V. Lepetit, “Speed invariant time surface for learning to detect corner points with event-based cameras,” in Proc. IEEE/ICCV Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 10237–10246.

[19] I. Alzugaray and M. Chi, “HASTE: multi-hypothesis asynchronous speeded-up tracking of events,” in Proc. British Machine Vision Conference (BMVC), 2020, pp. 744–754.

[20] ——, “ACE: An efficient asynchronous corner tracker for event cameras,” in Proc. International Conference on 3D Vision (3DV), 2018, pp. 653–661.

[21] H. Seok and J. Lim, “Robust feature tracking in DVS event stream using Bézier mapping,” in Proc. IEEE Winter Conference on Applications of Computer Vision (WACV), 2020, pp. 1647–1656.

[22] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza, “EKLT: Asynchronous photometric feature tracking using events and frames,” International Journal of Computer Vision (IJCV), vol. 128, no. 3, pp. 3867–3876, 2020.

[23] G. Gallego and D. Scaramuzza, “Accurate angular velocity estimation with an event camera,” IEEE Robotics and Automation Letters, vol. 2, no. 2, pp. 632–639, 2017.

[24] G. Gallego, H. Rebecq, and D. Scaramuzza, “A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation,” in Proc. IEEE/CFV Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 3867–3876.

[25] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in Proc. International Conference on Knowledge Discovery and Data Mining, 1996, pp. 226–231.

[26] E. Mueggler, H. Rebecq, G. Gallego, and D. Scaramuzza, “The event-camera dataset and simulator: event-based data for pose estimation, visual odometry, and SLAM,” International Journal of Robotics Research, vol. 36, no. 2, pp. 142–149, 2017.

[27] D. Gehrig, B. Kohn, M. Yang, S.-C. Liu, and T. Delbrück, “A 240 × 180 130 dB 3 μs latency global shutter spatiotemporal vision sensor,” IEEE Journal of Solid-State Circuits, vol. 49, no. 10, pp. 2333–2341, 2014.

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

[29] B. D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” in Proc. International Joint Conference on Artificial Intelligence (IJCAI), 1981, pp. 674–679.