COUNTING REPETITIVE ACTIONS IN EVENT STREAM

Yuelong Zhuo[1], Weiling Li[1], Beibei, Yang[1], Yan Fang[2], Huaqiang Yuan[1]

[1]School of Computer Science and Technology, Dongguan University of Technology, Dongguan, China
[2]Department of Electrical and Computer Engineering, Kennesaw State University, Marietta, GA, USA

ABSTRACT

The frame-based method is not suitable for counting repetitive actions in event stream, since the framing process will disrupt the temporal information of events. For accurate count of repetitive actions in events, we propose a framework based on threefold ideas: a) converting event stream into time series, b) searching candidates of repetitive actions based on the ascending and descending trends of event time series, and c) checking the candidates with a fast dynamic time warping based method. For accurate counting repetitive actions, an action enhancement method for event time series and a Mann-Kendall test incorporated dynamic candidate selection algorithm are innovatively proposed. The experimental results on artificially synthesized and normally recorded event datasets demonstrate that our framework can count repetitive actions in event stream with high accuracy. All codes, datasets and examples of visualization can be found at https://github.com/ZYL618/action_count_in_events3.

Index Terms—Repetitive Actions, Event Stream, Time Series, Mann-Kendall Test, Fast Dynamic Time Warping

1. INTRODUCTION

Repetitive actions are ubiquitous in nature phenomena and daily life. It is very important to detect and count repetitive actions since intents and driving causes are usually behind actions happened multiple times. Repetitive action counting in computer vision [1] is also useful for video analysis applications, such as pedestrian detection [2] and biological events measurement [3]. However, due to the limitations of traditional camera hardware and design principles, its ability to adapt to low light environment is relatively limited, and it is prone to motion blur or failure to capture objects with fast motion, i.e., it is easy to lead to the loss of image details in high dynamic range. It is very important to find a new way to deal with the repetitive action count in special scenes.

In recent years, event cameras have demonstrated the commercial value of their visual model. As a biologically inspired sensor, event cameras have a high level of temporal resolution at the microsecond level [4-5]. They detect brightness changes using a Dynamic Vision Sensor (DVS), which enables vision under poor lighting with quick responses [6]. Compared to conventional cameras, event cameras can more effectively record repetitive actions for its dynamic vision [4]. Therefore, by using event cameras, some repetitive action data that conventional cameras cannot record can be collected.

A system that can count repeating actions accurately will help us better analyze the information behind the actions. In order to achieve this goal, many methods have been devised. Dwibedi *et al*. proposed a network named RepNet [7] to learn similarity among frames with a temporal self-similarity matrix for repetitive action count. Hu *et al*. proposed TransRAC in [8], which encoding multi-scale temporal correlation with transformers for better performance. However, these methods can only receive frames as input, i.e., they cannot directly process event streams. Although aggregating events into frames by period is a commonly used operation for handling events [9], it is not feasible in the problem of counting repetitive actions since a) the correspondence between events and actions is unknown, b) The events in a frame have no order, which destroys the spatiotemporal correlation between events [10]. Hence, it is necessary to design a special model for the event stream data to count.

Recall that a DVS outputs an event when it is triggered by brightness change over a threshold, the output data of an event camera is a stream [5]. For counting repetitive actions in event stream, a framework is proposed in this paper. The first step of this framework is to convert event stream into multi-dimensional time series according to the spatial coordinate as shown in Fig. 1. Thus, turning visual-based tasks such as capturing key trails into time series processing tasks [11].

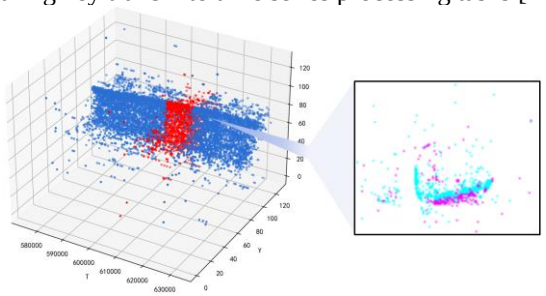
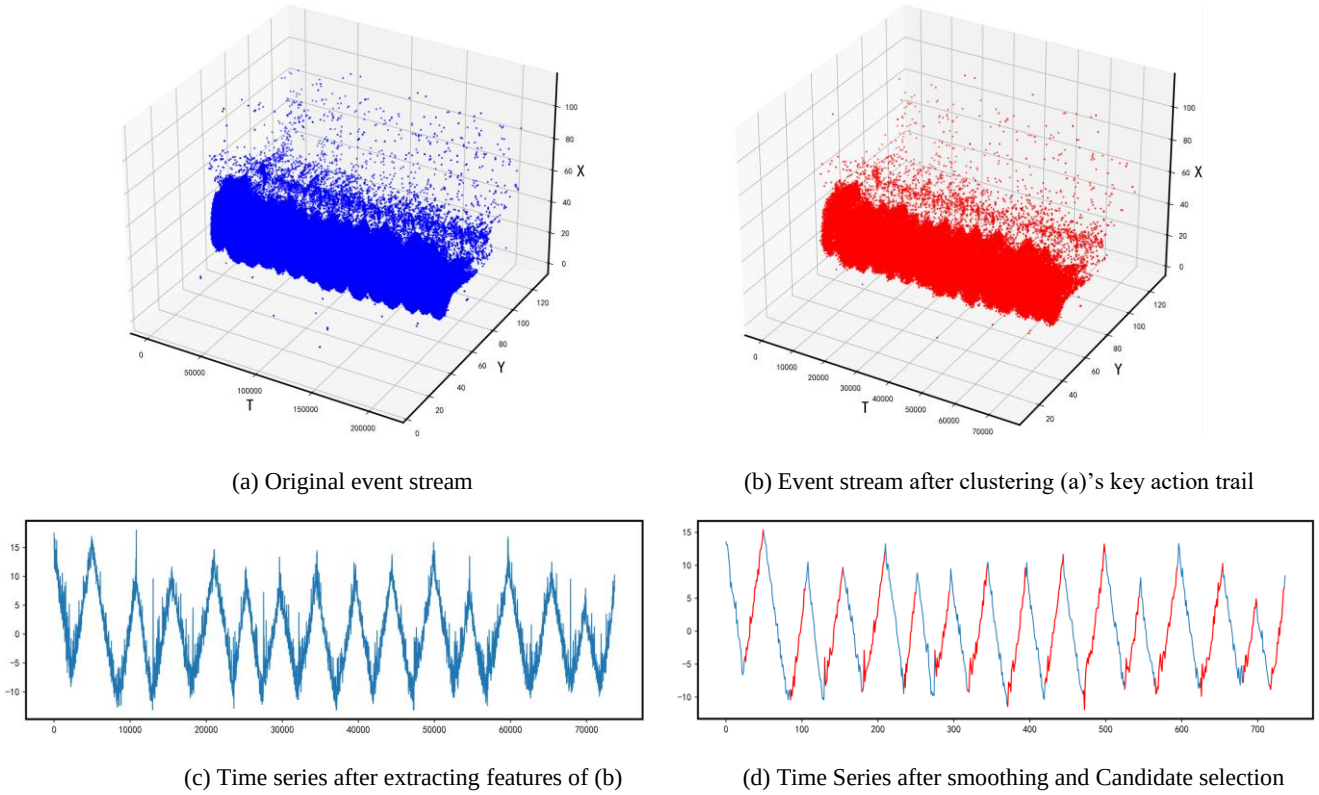


Fig. 1: Events stream is rendered in 3D space, where the red part is the event frame over a certain period of time.

To the authors best knowledge, this is the first work to consider counting repetitive actions in event stream obtained by DVS. The experimental results on artificially synthesized and normally recorded event datasets demonstrate that our



(a) Original event stream

(b) Event stream after clustering (a)'s key action trail

(c) Time series after extracting features of (b)

(d) Time Series after smoothing and Candidate selection

Fig. 2: The outputs of different modules in proposed framework.

framework can count repetitive actions in event stream with high accuracy.

2. PROPOSED FRAMEWORK

In this section, we provide the details of the proposed framework. The input of this framework is an event stream and it outputs the number of repetitions if there are repetitive actions recorded in the events. The framework consists of four modules, namely event time series initialization, action enhancement, candidate selection and repetitive action count. The main steps are shown in Fig. 2.

2.1. Event Time Series Initialization

This section provides how we transform event stream into time series.

Event stream. DVS outputs event stream when it is triggered by brightness change over the threshold. Each event in the stream contains a (x, y, t, p) 4-tuple, where t is the timestamp in microseconds, x and y are the pixel address and $p \in \{-1, 1\}$ is the polarity of the brightness change.

Transforming events to time series. By arranging events in timestamp order and reading out x and y of each event, we have a 3D time series, e.g., $T=\{(x_1, y_1), (x_2, y_2)\ldots, (x_i, y_i) \ldots, (x_n, y_n)\}$, where x_i and y_i represents the value of the i-th event in the x direction and the y direction, respectively. Note that we ignore the polarity since either polarity value indicates that an action has occurred at the current position.

2.2 Action Enhancement

The function of this module is twofold: a) filtering noise and identify key action trails by utilizing the spatiotemporal correlation of events, and b) extracting the features of the event stream, convert them into a time series and smooth them.

Clustering events of key action trail. We found that events corresponding to an action has strong spatiotemporal correlation [12]. Based on this fact, a spatiotemporal clustering method is designed, it works by fulfilling following steps:

a) Initialize n=1;

b) Initialize an empty set W of size C. Reading T, when i-th element is read and W is empty, $W=\{i\}$, we have a baseline index $\gamma=i$;

c) If both of the distance between x_i and x_γ and the distance between y_i and y_γ are not bigger than a set threshold, e.g., δ, the i-th event is considered to have strong spatiotemporal correlation with γ-th event, we add i to W;

d) Repeating c) until W is full or either the distance between x_i and x_γ and the distance between y_i and y_γ is bigger than δ. Then the average of $\{(x_j,y_j)\}$ for $j \in W$, e.g., $\{\hat{x}_n, \hat{y}_n\}$ is recorded by sequence $\hat{T}$;
e) Remove $\{(x_j,y_j)\}$ for $j \in W$ from T, and set $n=n+1$;
f) Repeat b) to e) until there is no elements in T.

Extracting features of key action trail. After the process of clustering events of key action trail, events with strong spatiotemporal correlation are clustered and stored in new sequences. To further analyze the event time series, it should be preprocessed firstly for dimensionality reduction. In this work, we utilized Principal Component Analysis (PCA) [13], which is a widely used method in data analysis and data dimensionality reduction, to perform feature extraction and time series transformation. Otherwise, time series can be extracted into two parts by Hodrick-Prescott (HP) filtering: cyclical component and trend component [14]. The cyclical component describes the short-term fluctuations or periodic changes of the time series, which reflect the short-term oscillations with respect to the trend. The trend component represents the long-term trend of the time series, that is, the smooth part of the data, which represents the underlying trend or long-term change of the time series. In the repetitive counting task, we pay more attention to the overall trend change of the time series, so the trend component is taken as the subsequent processing object

Specifically, we apply PCA to the sequence $\hat{T}$ and use HP filtering to obtain the cyclical component $\tilde{T} = \{\tilde{t}_1, \tilde{t}_2, ..., \tilde{t}_n\}$, which contains the features in the sequence $\hat{T}$ and has only two dimensions.

Smoothing key action trail. After action enhancement and extracting features of key action trail, the noise in the time series can be reduced, but the amount of data is still large, which is not conducive for trend analysis. Therefore, we use the simple moving average (SMA) [15], a common statistical method for smoothing time series data. SMA can smooth the short-term fluctuations in time series and highlight the potential trend. With SMA, the size of the data can be reduced and the trend information in the time series can be further highlighted.

Hence, we have new sequence:

$$\bar{T} = \{\bar{t}_1, \bar{t}_2, ..., \bar{t}_m\} = \{\frac{\sum_{i=1}^{\eta} \tilde{t}_i}{\eta}, \frac{\sum_{i=\eta+1}^{2\eta} \tilde{t}_i}{\eta}, .., \frac{\sum_{i=(m-1)\eta+1}^{n} \tilde{t}_i}{n-(m-1)\eta}\} \quad (1)$$

where η is the window size of SMA and $m=\lceil n/\eta \rceil$.

2.3 Candidate Selection

Since $\bar{T}$ contain motions of the target object over a period of time. Repetitive actions if existed are subsets of the motions. However, the length and position of repetitive actions are unknown. How to find candidates of repetitive actions from $\bar{T}$ is a challenging task since we cannot determine the candidate's length and position. Fortunately, the actions, which are the most concerned about in events, are composed of different changing trends in $\bar{T}$. Therefore, we can find candidates by segmenting $\bar{T}$ by noting the shift points of trends.

Although we have smoothed the time series, there is still vibration in $\bar{T}$, so we cannot simply make trend judgments based on the value of adjacent event points, but need to grasp the overall trend. The Mann-Kendall (MK) test [16-17] is a non-parametric statistical method commonly used to assess the presence of trends in time series data. It focuses on the relationship between adjacent observations in a time series, comparing their relative magnitudes to detect the presence of a trend and emphasizes the direction of trends rather than their specific magnitudes. However, directly using $\bar{T}$ as the input for MK test can only obtain the overall trend of $\bar{T}$. Hence, we proposed a dynamic candidate selection method which incorporates MK test into a dynamic framework for more accurate shift points. The detail of proposed algorithm is as follows:

Algorithm 1: Dynamic Candidate Selection

Input: time series X, window size w;
Initialization: $d=5$, sets of shift points $P=\{\}$ $Q=\{\}$;
Output: P;
1: P=SPD(X, w)
2: Q=SPD(X, $w+d$)
3: $v1$=var(diff(P))
4: $v2$=var(diff(Q)
5: **while** $v2<v1$ **do**
6: v1=v2;
7: $P=Q$;
8: $w=w+d$;
9: Q=SPD(X, $w+d$);
10: $v2$=var(diff(Q));
11: **end while**

*var() and diff() are Python commands provided by Numpy.

Procedure 1: Shift Point Detection (SPD)

Input: time series X, window size w;
Initialization: $i=1$, stepsize s, *maximum number of segmentation* $m=(|X|-w)/s$, a set of shift points $G=\{\}$;
Output: G
1: **while** $i<m$ **do**
2: $t1$=MK($X\{(i-1)\times s+1: (i-1)\times s+w\}$)
3: $t2$=MK($X\{i\times s+1: i\times s+w\}$)
4: ***if*** $t1!=t2$:
5: insert $\lceil (i-1)\times s+w)/2 \rceil$ to G
6: **end while**

*MK(X) means doing MK test on X, which outputs 1, 0, -1 for upward trend, no trend, and a downward trend, respectively.
*$X\{a: b\}$ denotes a piece of X from x_a to x_b.

2.4 Repetitive Action Count

Based on the shift points output by algorithm 1, we can segment $\bar{T}$ into multiple pieces, each of them is a candidate.

According to the type of trends, i.e., the ascending and descending trends, candidates can be divided into two sets, e.g., *A* and *D*. To count repetitive actions with these candidates, we should evaluate the similarity, which can be quantified by matching cost, of any two candidates in one set. Because repetitive actions may occur with slight variations, dynamic time warping (DTW) [18-20] is considered to be a solution to calculate the matching cost of time series. It finds an optimal alignment path between two time series by minimizing their overall distance. When comparing two sequences, DTW allows for local stretching or compressing of each sequence in the time dimension to achieve the best match. However, the traditional DTW algorithm has a quadratic time complexity, which can be computationally expensive for long time series. Fast DTW addresses this limitation by introducing a lower-bound constraint [21], allowing it to significantly reduce the computation time while providing a reasonably accurate approximation of the DTW distance.

With fast DTW, we can calculate the normalized alignment cost, e.g., r_{ij}, of any two candidates i and j, in one set, and store them in matrix R. By taking a row-wise average on R, we have a vector v, where v_i represents the average of r_i. By this operation, similar candidates will have relatively close matching costs, while the matching costs corresponding to irrelevant candidates will be further separated. Then we sort v_i sequentially. When the difference between v_i and v_{i+1} is higher than the threshold value of σ, argues that i candidates are considered to be similar, i.e., there are i repetitive actions on target events.

3. EXPERIMENT

3.1. Experiment Settings

Datasets. In the experiment, we use a public dataset named DvsGesture [22]. For testing the performance of proposed framework. Table 1 lists the details of datasets used in the experiment. For involved datasets, the ground truth repetition count is the count of the most repeated action.

Table 1. Details of used datasets

Dataset	Description
D1	It contains five types of gestures displayed by the same individual under five types of lighting conditions. They are randomly obtained from DvsGesture dataset.
D2	It is an artificially synthesized dataset. The data is obtained by repeatedly synthesizing a single action intercepted from D1.
D3	It is an artificially synthesized dataset. The data is obtained by synthesizing another action intercepted from D1 before and after D2.
D4	It is an artificially synthesized dataset. The data is obtained by inserting another action intercepted from D1 into D2 while remaining actions' integrity.

Datasets Preprocessing. We use inivation DV to convert event stream files from aedat files to csv files, which records the timestamp, pixel address and polarity of events. The data storage format of event-driven vision sensor is AEDAT file instead of traditional frame data. CSV file is a common text file format for storing tabular data. By using inivation DV, we can extract and store the timestamp, coordinates and polarity information of event points in AEDAT and use CSV for further processing. In the process of conversion, the Dvsnoisefilter module is introduced to filter the noise initially, which is the built-in event stream noise filtering algorithm of DV. For frame-based repetitive action count models, we generate frame videos based on D1-D4. Events in D1 are converted to videos of 30FPS. Events in D2-D4 are converted to videos of 15FPS, where each frame contains 2000 events.

Comparison model. REPNET [7] is used as a comparative model in our experiment. It takes a video as input and output count of repetitive actions.

Evaluation Metrics: two matrices, e.g., Off-By-One (OBO) count error and Mean Absolute Error (MAE), are used to evaluate the count accuracy of involved methods [8].

OBO count error is a specific indicator for counting repetitive actions. It is based on the idea that if the predicted count is within one count of the ground truth, we can consider counting is correctly and can be calculated as follows:

$$OBO = \frac{1}{N}\sum_{i=1}^{N}[|c_i - \tilde{c}_i| \le 1]. \quad (2)$$

where $\tilde{c}$ is the ground truth repetition count. N is the number of given datasets.

MAE means normalized absolute error between the ground truth count and the predicted count, which can be calculated as follows:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\frac{|c_i - \tilde{c}_i|}{\tilde{c}_i}, \quad (3)$$

Implementation Settings. All experiments are implemented on a platform with a CORE-i5 2.50 GHz CPU and RTX 2050 GPU. The proposed framework is implemented with Python 3.11.4 and RepNet is implemented with Tensorflow.

3.2. Experimental Results

Based on the experiment settings, experiments have been taken to evaluate the performance of proposed framework. The experimental results are listed in Tables 2-4. From the experimental results, we have following findings:

a) **The proposed framework has a strong ability in counting natural repetitive actions in events obtained in different light conditions.** As shown in Table 2, the proposed method obtains a very high OBO and low MAE values in all cases on D1. Especially, under the first four lighting conditions, the proposed model can accurately count repetitive actions. Under the last lighting condition, it can also achieve high counting

accuracy. Moreover, the proposed framework outperforms RepNet in all cases on D1. It indicates that our method outperforms the frame-based counting model in counting repetitive actions in events.

b) **Directly using events for counting repetitive actions can avoid the impact of framing operations on the results.** As shown in Table 3, for D2 event data synthesized by a single action, both proposed framework and RepNet achieve the highest OBO count error on it. However, the proposed framework outperforms RepNet on MAE. Furthermore, MAE equals 0 means that our counting results are completely correct. The experimental results indicate that frame based pre training model will be susceptible to inaccurate interference from event frame aggregation.

Table 2. MAEs and OBOs obtained by involved methods on D1

Type of lights	Metrics	Models	
		RepNet	**Ours**
fluorescent	MAE↓	0.280	**0.000**
	OBO↑	0.200	**1.000**
fluorescent_led	MAE↓	0.263	**0.042**
	OBO↑	0.200	**1.000**
lab	MAE↓	0.319	**0.000**
	OBO↑	0.400	**1.000**
led	MAE↓	0.283	**0.022**
	OBO↑	0.400	**1.000**
natural	MAE↓	0.256	**0.079**
	OBO↑	0.600	**0.800**

Table 3. MAEs and OBOs obtained by involved methods on D2

Metrics	OBO↑	MAE↓
RepNet	1.000	0.200
Ours	**1.000**	**0.062**

c) **The proposed model has the potential to identify multiple repetitive actions from events and count them.** As shown in Table 4, although D3 and D4 are events synthesized from multiple repetitive actions, the proposed model still achieves high counting accuracy on them. It indicates that our model can accurately identify the actions with the highest number of repetitions from events. From section 2.4, we can see that further improvement of the framework can distinguish different actions from complex event data and count them.

Table 4. MAEs and OBOs obtained by involved methods

Dataset	Metrics	Models	
		RepNet	**Ours**
D3	MAE↓	0.273	**0.130**
	OBO↑	0.000	**0.500**
D4	MAE↓	0.346	**0.097**
	OBO↑	0.167	**0.833**

3.3. Summary

We summarize the pros and cons of proposed framework based on the experimental results:

Pros. The proposed framework transforms event stream into time series, so potential repetitive actions can be identified and counted by analyzing time series. Since it does not need to aggregate events into frames, it avoids the action segmentation error caused by the aggregation process. Additionally, this method is purely mathematical and does not require pre training, which has significant advantages in complexity compared to neural network-based counting models.

Cons. The proposed framework is currently only applicable to single target repetitive actions and cannot handle multiple target repetitive actions. Moreover, the combination of hyper-parameters has an influence on the performance of proposed method.

4. CONCLUSION

In this work, we propose a framework for counting repetitive actions in events. It firstly transmits events to time series, by doing that, the repetition count problem can be converted to a time series analysis task. For accurate counting repetitive actions, an action enhancement method for event time series and a Mann-Kendall test incorporated dynamic candidate selection algorithm are innovatively proposed. The experimental results on artificially synthesized and normally recorded event datasets demonstrate that our framework can count repetitive actions in event stream with high accuracy.

Acknowledgement: This work was supported in part by the National Natural Science Foundation of China under grant 62102086, in part by the Guangdong Basic and Applied Basic Research Foundation under grant 2021B1515140046 and in part by NSF of United State under grant 2153440. Weiling Li is the corresponding author of this paper.

5. REFERENCES

[1] T. F. H. Runia, C. G. M. Snoek, and A. W. M. Smeulders, "RealWorld Repetition Estimation by Div, Grad and Curl." arXiv, Feb. 27, 2018.

[2] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL: IEEE, Jun. 2009, pp. 304–311.

[3] S. Lee and D. Park, "A Real-Time Abnormal Beat Detection Method Using a Template Cluster for the ECG Diagnosis of IoT Devices," *Human-centric Computing and Information Sciences*, vol. 1, no. 0, pp. 1–15, Jan. 2021.

[4] G. Gallego *et al.*, "Event-Based Vision: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 154–180, Jan. 2022.

[5] X. Wang *et al.*, “VisEvent: Reliable Object Tracking via Collaboration of Frame and Event Flows.” arXiv, Jun. 28, 2022.

[6] P. Lichtsteiner, C. Posch, and T. Delbruck, “A 128×128 120 dB 15 μs latency asynchronous temporal contrast vision sensor,” IEEE J. SolidState Circuits, vol. 43, no. 2, pp. 566–576, 2008.

[7] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman, “Counting Out Time: Class Agnostic Video Repetition Counting in the Wild.” arXiv, Jun. 27, 2020.

[8] H. Hu, S. Dong, Y. Zhao, D. Lian, Z. Li, and S. Gao, “TransRAC: Encoding Multi-scale Temporal Correlation with Transformers for Repetitive Action Counting,” in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, LA, USA: IEEE, Jun. 2022, pp. 18991–19000.

[9] 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 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT: IEEE, Jun. 2018, pp. 3867–3876.

[10] H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza, “High Speed and High Dynamic Range Video with an Event Camera,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 6, pp. 1964–1980, Jun. 2021.

[11] Q. Wen, K. He, L. Sun, Y. Zhang, M. Ke, and H. Xu, “RobustPeriod: Time-Frequency Mining for Robust Multiple Periodicity Detection,” in *Proceedings of the 2021 International Conference on Management of Data*, Jun. 2021, pp. 2328–2337.

[12] S. Guo and T. Delbruck, "Low Cost and Latency Event Camera Background Activity Denoising," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 45, no. 1, pp. 785-795, 1 Jan. 2023.

[13] H. Li, “Asynchronism-based principal component analysis for time series data mining,” Expert Systems with Applications, vol. 41, pp. 2842–2850, May 2014.

[14] R. J. Hodrick and E. C. Prescott, “Postwar U.S. Business Cycles: An Empirical Investigation,” Journal of Money, Credit and Banking, vol. 29, no. 1, p. 1, Feb. 1997.

[15] M. Yudono *et al.*, “Bitcoin USD Closing Price (BTCUSD) Comparison Using Simple Moving Average And Ra-dial Basis Function Neural Network Methods,” *FIDELITY : Jurnal Teknik Elektro*, vol. 4, pp. 29–34, May 2022.

[16] M. Sayemuzzaman and M. K. Jha, “Seasonal and annual precipitation time series trend analysis in North Carolina, United States,” *Atmospheric Research*, vol. 137, pp. 183–194, Feb. 2014.

[17] M. Gocic and S. Trajkovic, “Analysis of changes in meteorological variables using Mann-Kendall and Sen’s slope estimator statistical tests in Serbia,” *Global and Planetary Change*, vol. 100, pp. 172–182, Jan. 2013.

[18] P. Senin, “Dynamic Time Warping Algorithm Review,” Jan. 2009.

[19] C. Souza, C. Pantoja, and F. C. Souza, “VERIFICAÇÃO DE ASSINATURAS OFFLINE UTILIZANDO DYNAMIC TIME WARPING,” Dec. 2009, pp. 1–5.

[20] A. Corradini, “Dynamic time warping for off-line recognition of a small gesture vocabulary,” in Proceedings IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, Vancouver, BC, Canada: IEEE Comput. Soc, 2001, pp. 82–89.

[21] S. Salvador and P. Chan, “Toward accurate dynamic time warping in linear time and space,” IDA, vol. 11, no. 5, pp. 561–580, Oct. 2007.

[22] A. Amir *et al.*, “A Low Power, Fully Event-Based Gesture Recognition System,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI: IEEE, Jul. 2017, pp. 7388–7397.