IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 17, 2021, accepted May 23, 2021, date of publication June 3, 2021, date of current version June 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3085708

Making Sense of Neuromorphic Event Data for Human Action Recognition

SALAH AL-OBAIDI, HIBA AL-KHAFAJI, AND CHARITH ABHAYARATNE, (Member, IEEE)
Department of Electronic and Electrical Engineering, The University of Sheffield, Sheffield S1 3JD, U.K.
Corresponding author: Charith Abhayaratne (c.abhayaratne@sheffield.ac.uk)

ABSTRACT Neuromorphic vision sensors provide low power sensing and capture salient spatial-temporal events. The majority of the existing neuromorphic sensing work focus on object detection. However, since they only record the events, they provide an efficient signal domain for privacy aware surveillance tasks. This paper explores how the neuromorphic vision sensor data streams can be analysed for human action recognition, which is a challenging application. The proposed method is based on handcrafted features. It consists of a pre-processing step for removing the noisy events followed by the extraction of handcrafted local and global feature vectors corresponding to the underlying human action. The local features are extracted considering a set of high-order descriptive statistics from the spatio-temporal events in a time window slice, while the global features are extracted by considering the frequencies of occurrences of the temporal event sequences. Then, low complexity classifiers, such as, support vector machines (SVM) and K-Nearest Neighbours (KNNs), are trained using these feature vectors. The proposed method evaluation uses three groups of datasets: Emulator-based, re-recording-based and native NVS-based. The proposed method has outperformed the existing methods in terms of human action recognition accuracy rates by 0.54%, 19.3%, and 25.61% for E-KTH, E-UCF11 and E-HMDB51 datasets, respectively. This paper also reports results for three further datasets: E-UCF50, R-UCF50, and N-Actions, which are reported for the first time for human action recognition on neuromorphic vision sensor domain.

INDEX TERMS Neuromorphic vision sensing (NVS), event cameras, dynamic vision sensing (DVS), human action recognition (HAR), local features, global features.

I. INTRODUCTION

Neuromorphic vision sensing (NVS), also known as dynamic vision sensing and event camera sensing, which has emerged recently, is capable of capturing fast spatio-temporal spikes (changes) in a scene with low power consumption [1]–[8]. Such data is of the form of a continuous stream of spatio-temporal *events* or *spikes*, as opposed to regularly uniformly spatio-temporal sampled values traditions imaging systems, as in active pixel sensing (APS). This allows NVS to measure changes in intensity at each pixel asynchronously, instead of acquiring the intensity of that pixel, *i.e.*, non-uniformly sampling temporally leading to a rendering frame rate up to 2000 fps with consuming low power. It encodes the intensity change at each pixel in the form of an *event* or a *spike*. FIGURE 1 shows an example of a stream of events for a person running. This stream is represented as

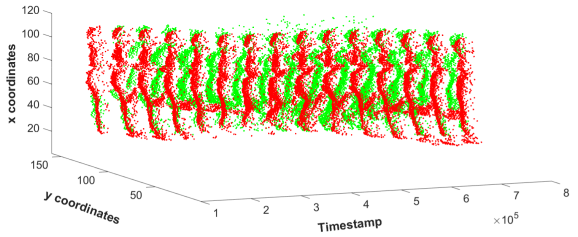


FIGURE 1. Representation of the events of a running action using an emulator to generate the events. Green/Red points are for visualisation of ON and OFF events.

3D points, referring to the spatial location (in terms of (x,y) coordinates and the time of the event. An event is recorded either as an initiation (ON) and a termination (OFF), shown in green and red points, respectively in FIGURE 1. Each event in this figure has potentially valid information that can be explored to understand the scene, in terms of object and action recognition. Although success has been reported in visual content understanding using traditional imaging in the

The associate editor coordinating the review of this manuscript and approving it for publication was Alessia Saggese.

literature, in order to optimally use event or spike data from NVS, novel algorithms for processing and learning such data are needed. In this work, we explore how NVS data can be analysed for human action recognition (HAR).

HAR from video sequences captured using conventional APS imaging systems primarily detect and model motion patterns to learn important features of an action to train a classifier [9]–[22]. The features can be either handcrafted or learned by deep learning approaches. Although these have shown very high accuracy rates for benchmark datasets, such conventional vision systems often suffer from many limitations, such as, limited frame rate, high redundancy within the successive frames and motion blurring, affecting the performance of action recognition [22], [23]. Also, pre-processing steps, such as estimating motion from video pixels (block matching, optic flow or phase correlation) are computationally expensive. Furthermore, conventional video-based HAR has also caused privacy issues in the context of assisted living [24]–[26]. Exploration of NVS data for HAR also enables to overcome some of these limitations intrinsic to conventional imaging-based HAR. As NVS encodes the intensity change at each pixel and samples at non-uniform sampling rates, events with high frequency of occurrences correspond to high motion present in the scene, which is a solution for motion blurring due to high speed motion as often seen in conventional APS cameras. Such a high motion response means that NVS based camera is regarded as a data-driven sensor since the output NVS depends on the magnitude of the apparent motion in the scene [23]. These advantages combined with low power consumption and low throughput for streaming have emerged NVS as a suitable vision sensor for robotics and mobile-based applications [27]–[29].

Although NVS-based vision applications have seen emerged fast recently [23], it has not resulted in many works in human motion analysis. Most recent works exploring NVS data for human motion analysis consists of low semantic tasks, such as, hand or finger movement analysis [30]–[37] and human fall detection [38]. However, exploring NVS data for higher-level semantic tasks, such as, multi-class HAR, is still in early stages [39]–[43]. One reason for this slow progress of NVS domain HAR is the high cost of NVS devices compared to the conventional APS cameras [8] leading to insufficient annotated NVS domain HAR training datasets [41], [44]. Recently emerged software-based emulators for converting APS data into NVS data [45], [46] were also found useful for generating test data.

However, rather than extending conventional HAR approaches used in classical computer vision, new paradigms are need to be explored for efficiently understanding NVS data for HAR and other applications. Some of the challenges in NVS data include presence of noisy events, understanding true motion, lack of clarity of contexts in object boundaries due to lack of intensity data, high sparsity of data and missing spatio-temporal connectivity in NVS data. Therefore, effective NVS-domain feature extraction algorithms are needed for the advancement of usage of NVS devices in real applications. In this work, we present a novel methodology for efficient understanding of NVS data for HAR applications. The main contributions of our work include:

1) A methodology for pre-processing NVS data including a new algorithm for de-noising NVS data, *i.e.*, to remove the noisy events that may have been resulted in due to certain acquisition parameters used in NVS devices;
2) A methodology for extracting a new set of global temporal features to model the global (long term) motion patterns considering a long duration NVS event data stream;
3) A methodology for extracting a new set of local spatio-temporal features to model local (short term) motion patterns considering a shorter durations of connected events in short durations of an NVS event data stream; and
4) Fusion of features for training a classifier and evaluation of the proposed method for various types of NVS data (real data, emulated data and recorded NVS from an RGB playback) covering various types of actions.

The rest of this paper is organized as follows: Section II reviews the related work on exploring the NVS domain for HAR. In Section III, we present the proposed methodology for understanding NVS data for HAR. Section IV shows the experimental evaluation of the performance of the proposed method and discussion followed by the concluding remarks in Section V.

II. RELATED WORK

The existing work on neuromorphic vision sensing in computer vision can be grouped into three themes: object detection [47]–[49], pedestrian detection [50], [51] and hand gesture recognition [33]–[35]. There is only a little work on exploring the neuromorphic data beyond object detection addressing highly semantic applications, such as, multi class action recognition, which still poses an important challenge. As mentioned in Section I, work on using NVS data for HAR is still in early stages [39]–[43]. Most of these methods start with temporally aggregating the polarities into a collection of NVS data frames by considering a non-overlapping time window corresponding to the frame rate of conventional APS cameras. This is followed by using these NVS frames for either extracting handcrafted motion features or learned features for representing the actions in the test sequences.

In [41], 8-bit gray-scale frames are constructed from the events. Pixels of these frames are initialised with 128 and then either are increased or decreased considering the polarities of the events recorded at each spatial location (pixel) by considering the time interval corresponding to an actual frame. This is followed by extracting motion event features (magnitude and direction of motion) considering stacked event frames with variable stack sizes depending on the

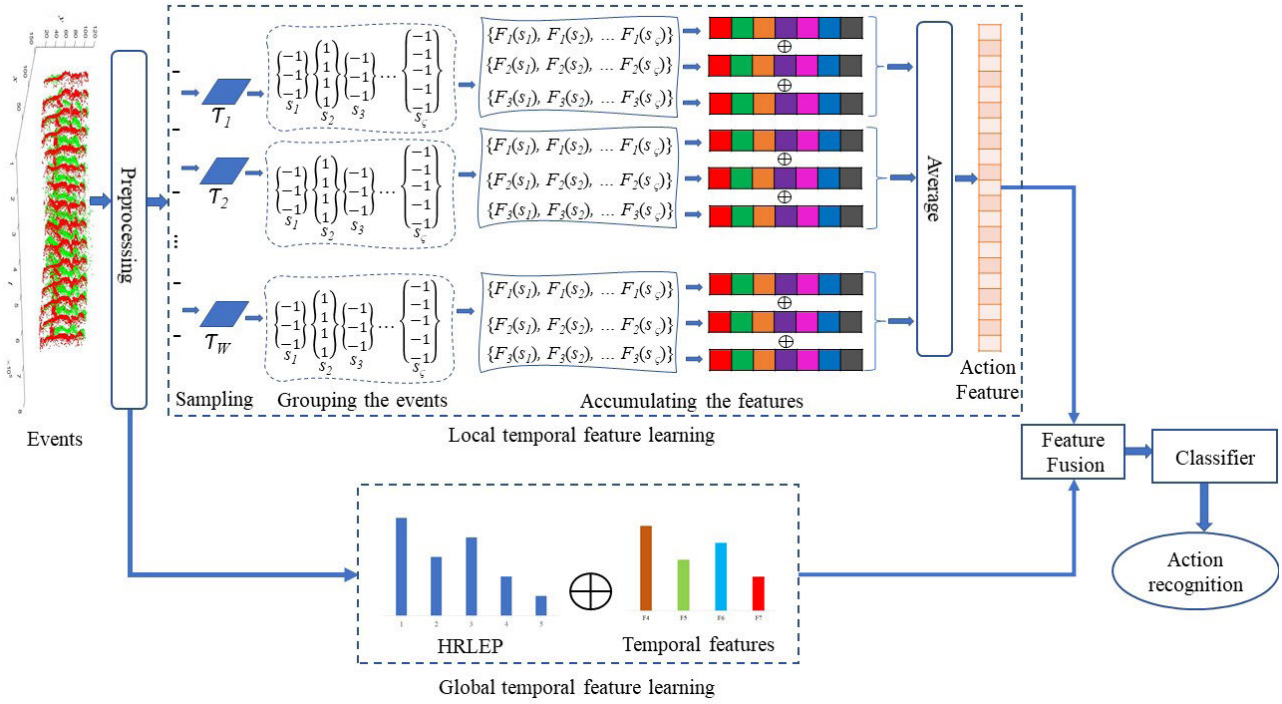


FIGURE 2. The pipeline of the proposed method for NVS domain HAR.

motion level in the activity. Finally, these motion information is fed into a convolution neural network (CNN) for feature learning for HAR. They demonstrate the usability of event data in HAR compared to conventional camera-based vision systems, where complex optic flow estimation is required. A similar approach was followed in [40] by using two CNNs to learn features from event frames and corresponding optic flow from the original RGB. Converting the events into frames is also applied in [43] to classify the actions of the neuromorphic version of UCF11 dataset. A time stamp aggregation algorithm is used to create the frames from the events, where these frames are fed into CNN for classification with a 92.90% of accuracy. In [39], three 2D motion maps (on x-y, x-z and y-z planes) and Motion Boundary Histogram (MBH) are constructed from the events. Speeded Up Robust Features (SURF) are extracted through grid search on the 2D motion maps followed by k-means clustering to create a Bag of visual vocabulary (BoVV) of k words from motion maps and MBH. Finally, the feature vectors constructed from BoVV are used to train the linear SVM. Also Graph CNN based methods were reported for NVS-domain object recognition, with case studies on HAR in [42].

The existing work that uses hand-crafted features has achieved an accuracy rate of 75.13% [39], while the works that have used deep learning have achieved accuracy rates ranging from 51.5% to 92.9% [40]–[43] as detailed in Section IV-B. It can be also observed that the accuracy rates of these methods depend on the quality of the constructed event frames. The choice of time intervals plays a significant role in this. All these methods create motion maps from NVS events followed by either handcrafted feature learning or deep learning. In either way, they do not take the full advantage of NVS event data, which can be considered as motion information. Following the approach of frame creation or motion parameter estimation has added complexity similar to conventional cameras based vision algorithms. Therefore, in our present work, we focus on extracting features on the NVS event domain, *i.e.*, exploring the events directly, for HAR. Accordingly, we propose a new method that explores the NVS domain alone by considering the temporal patterns of ON and OFF events locally and globally to extract robust description for HAR. The proposed method analyses the patterns of the polarities using only NVS domain events and avoids converting the events into other domains without losing the essence of neuromorphic computing.

III. THE PROPOSED NVS DOMAIN FEATURE LEARNING

This section presents the proposed method for NVS domain HAR including the novel contributions on noise removal and constructing local and global spatio temporal event descriptors. FIGURE 2 depicts the block diagram of the proposed method with the pipeline of operations. The proposed method is divided into five main steps: pre-processing for noisy event removal, NVS domain local feature extraction, NVS domain global feature extraction, feature fusing, and classification. We start this section by introducing the NVS operation and the notation followed by the description of the main steps of the proposed method in subsequent subsections.

A. NVS OPERATION AND OUR NOTATIONS

In contrast to the standard pixel-domain based camera, where the sensors record the information of pixels at a constant frame rate, the NVS acquires the change of luminance with a variable sampling rate at each pixel. Accordingly, an event is triggered if the luminance at a pixel changes, *i.e.*, the log intensity exceeding a predefined threshold is sufficient to be considered as an event. This mechanism is performed independently and continuously for each pixel in the chip's array in NVS cameras, and the pixel is set to idle in case there is no luminance change has been detected, leading to temporally and spatially adapted independent and non-uniform temporal sampling for each pixel.

We denote an event, e_k, acquired at the coordinates, x_k and y_k, corresponding to a pixel, P_k, in the sensor array and at the timestamp, t_k, with the polarity p_k, *i.e.*, the orientation of the shifted log intensity, $\mathcal{L}P_k = \log(I_k)$, where k is the event index and I_k is the intensity at P_k. Thus, an event is represented as $e_k = (x_k, y_k, t_k, p_k)$ as soon as the magnitude of $\mathcal{L}(P_k)$ is shifted since the last event recorded at P_k, *i.e.*,

$$\Delta\mathcal{L}(x_k, y_k, t_k) = \mathcal{L}(x_k, y_k, t_k) - \mathcal{L}(x_k, y_k, t_k - \Delta t), \quad (1)$$

exceeds a temporal contrast threshold [7]. Δt is the time when the pixel P_k is idle since the last event at P_k. When the log intensity at P_k exceeds the, e_k is triggered with the polarity, $p_k \in \{-1, 1\}$, *i.e.*, the orientation of log intensity change, $\Delta\mathcal{L}$. It can be noticed that Eq. (1) is similar to finding the pixel difference between successive frames in conventional cameras based computer vision. This pixel difference, *i.e.*, log intensity, is evidence of the presence of motion in the scene. Therefore, this allows us to infer the implied motion in the scene by exploiting the events statistics rather than going through computationally expensive motion estimation algorithms often used in computer vision applications.

B. PRE-PROCESSING THE NOISY EVENTS

Depending on the threshold magnitude, some events are recorded in isolation without leading to any semantic meaning. We denote such events as noisy events and a pre-processing step for removing such events (de-noising) is applied on the events stream.

Let $\mathbb{E} = \{e_n | e_n = (x_n, y_n, t_n, p_n), \text{ and } 1 \leq n \leq N\}$, is a stream of events, where N is the length of the event stream. $\mathbb{E}$ is partitioned into time slices, $\mathbb{T} = \{\mathcal{T}_w | 1 \leq w \leq W\}$, where $\mathcal{T}_w$ is the time slice w. This partitioning is based on the principle of the frame rate that one would expect for a conventional camera video sequence. For example, if we have an NVS stream for 5 seconds, we generate 150 event slices assuming a 30 frames per second frame rate.

After partitioning the stream into event slices, for each slice let $\mathbf{E}_w = \{e_\ell | e_\ell = (x_\ell, y_\ell, t_\ell, p_\ell), \text{ and } 1 \leq \ell \leq L\}$, be the event stream in slice w, where L is the length of the total event stream in a slice, the following operations are applied. For each event e_ℓ at spatio-temporal location (x_ℓ, y_ℓ, t_ℓ), a 3×3 window on xy plane centered on the event location (x_ℓ, y_ℓ, t_ℓ) is considered and the number of events $C_{\ell_{(x,y)}}$ recorded on each of nine spatial coordinates (x, y) of the window over the total time of the slice is counted. This is followed by computing the total number of events in the 3D window-slice, S_ℓ, and the maximum events over the slice length, m_ℓ, as follows:

$$S_\ell = \sum_{i=x_\ell-1}^{x_\ell+1} \sum_{j=y_\ell-1}^{y_\ell+1} C_{\ell_{(i,j)}}. \quad (2)$$

$$m_\ell = \max_{i=x_\ell-1, j=y_\ell-1}^{i=x_\ell+1, j=y_\ell+1} C_{\ell_{(i,j)}}. \quad (3)$$

Finally, e_ℓ is processed to obtain new polarity, p'_ℓ, of the event as follows:

$$p'_\ell = \begin{cases} p_\ell & \text{if } S_\ell < (k \times 3 \times 3 \times m_\ell), \\ 0 & \text{otherwise,} \end{cases} \quad (4)$$

where $\{k \in \mathbb{R}^+ | k < 1\}$ is a user defined parameter for controlling the number of events to be removed. We present a discussion on the choice of the parameter k in Section IV-A.

C. LOCAL SPATIO-TEMPORAL FEATURE EXTRACTION

An action event stream can be represented considering the overall spatio-temporal patterns appear in the overall action sequence, as well as considering the local variations corresponding to the actions. In this section we address how to extract local features from the events stream, considering the events in partitioned time slices, $\mathcal{T}_w$. Since each action results in different spatio-temporal patterns of events at each time window, the local descriptors aim to recognise these patterns leading to representing discriminating features for specific action streams.

The process is started with $\mathbf{E}_w$ at $\mathcal{T}_w$, by sorting all e_ℓ in the ascending order of the x coordinate followed by grouping these events in $\mathcal{T}_w$ into $\{s_g | 1 \leq g \leq G\}$, where s_g defines ρ events that are successive and have the same polarity, such that,

$$s_g = \{e_i | e_i = (x_i, y_i, t_i, p_i), \text{ and } 1 \leq i \leq \rho\}, \quad (5)$$

where $x_{i+1} \geq x_i$ and $p_{i+1} = p_i \ \forall i$. According to Eq. (5), all events in s_g represent a pattern of log intensity change. Processing such patterns of polarities contributes to tracking the dynamic changes for each action and capturing the local structure of the events. This is achieved by modelling these changes in terms the relationship of horizontal and vertical locations, *i.e.*, (x, y) coordinates of the events in each set, s_g in terms of the following quantities:

$$m_g = \mu_x(s_g) - \mu_y(s_g), \quad (6)$$

$$v_g = \sigma_x^2(s_g) - \sigma_y^2(s_g), \quad (7)$$

$$d_g = \sigma_x(s_g) - \sigma_y(s_g), \quad (8)$$

where μ, σ^2 and σ are the mean, variance and the standard deviation of the spatial coordinates x and y of the events in s_g, respectively. This gives us three data vectors, $\mathbf{M}_w = \{m_g | 1 \leq g \leq G\}$, $\mathbf{V}_w = \{v_g | 1 \leq g \leq G\}$ and $\mathbf{D}_w = \{m_d | 1 \leq g \leq G\}$, for each $\mathcal{T}_w$. Then these data vectors are transformed

Algorithm 1 RLE of Polarities in a Stream of Events With N Events

```
Initialize Count ⟵ 0.
Initialize RunLengths ⟵ [ ].
for do i ⟵ 1 to N
    if p_i = p_{i+1} then
        Count ⟵ Count + 1,
    else
        RunLengths ⟵ [RunLengths  Count].
        Count ⟵ 0.
    end if
end for
Return RunLengths.
```

into 3 vectors containing higher order statistics of the data vectors as follows:

$$F_{1_w} = [\mu(\mathbf{M}_w), \max(\mathbf{M}_w), \min(\mathbf{M}_w), \sigma(\mathbf{M}_w), \ldots \ldots \sigma^2(\mathbf{M}_w), \gamma(\mathbf{M}_w), \kappa(\mathbf{M}_w)], \quad (9)$$

$$F_{2_w} = [\mu(\mathbf{V}_w), \max(\mathbf{V}_w), \min(\mathbf{V}_w), \sigma(\mathbf{V}_w), \ldots \ldots \sigma^2(\mathbf{V}_w), \gamma(\mathbf{V}_w), \kappa(\mathbf{V}_w)], \quad (10)$$

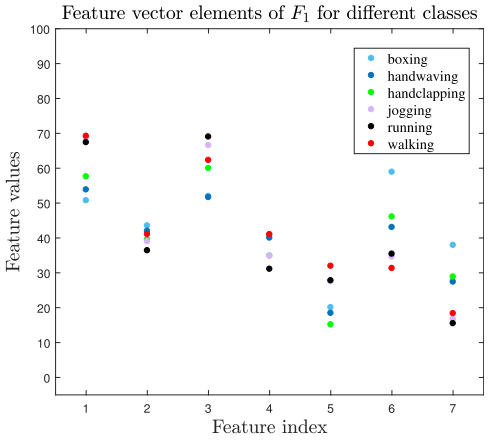
$$F_{3_w} = [\mu(\mathbf{D}_w), \max(\mathbf{D}_w), \min(\mathbf{D}_w), \sigma(\mathbf{D}_w), \ldots \ldots \sigma^2(\mathbf{D}_w), \gamma(\mathbf{D}_w), \kappa(\mathbf{D}_w)], \quad (11)$$

where γ and κ denote the skewness and the kurtosis, respectively. Then for each element in feature vectors, F_{1_w}, F_{2_w} and F_{3_w} the average over all W slices are computed to get the average feature vectors, F_1, F_2 and F_3, respectively. These three vectors are concatenated to get the local feature vector, $\mathcal{F}_L = \{F_1, F_2, F_3\}$, with 21 feature elements for the event stream $\mathbb{E}$. As an example, mean values of these feature vector elements for six sequences of one of the datasets (E-KTH) in FIGURE 3.

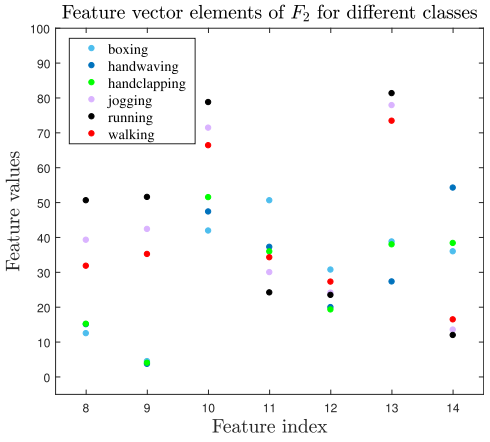
D. GLOBAL FEATURE EXTRACTION

Global features are extracted by considering the event stream for an action as a whole without resorting it into time-based slices. On the spatio-temporal event space, for each spatial coordinate (x, y), all temporal events are stacked into temporal groups, $\mathcal{H}_{\mathbb{E}} = \{\delta_h | 1 \leq h \leq H\}$, where H is the total number of temporal groups for the given (x, y). A group is defined as the continuous occurrence of events (either $p_l = +1$ or $p_l = -1$) at user-specified temporal sampling periods. The minimum events for a group is considered as 2, while just the isolated single events are disregarded as noise. For all events in δ_h, the consecutive similar polarity counts recorded as run-length encoding (RLE) as detailed in Algorithm 1. RLE keeps only the counts of consecutive occurrences without keeping the magnitudes of the polarities. Run lengths of all $\mathcal{H}_{\mathbb{E}}$ for all spatial locations are collected as a set, $\mathbb{R}$.

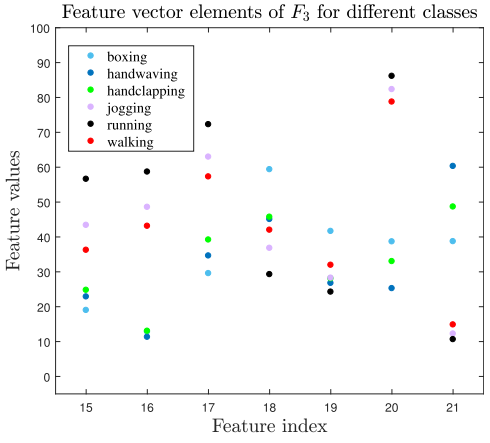
The first part of the global feature vector represents the spatial locations, $\mathbb{R}$, by computing the histogram of run-length encoded polarities (HRLEP), $\mathbb{H}$. Our experiments have found that partitioning HRLEP into 5 bins is sufficient to capture



(a) F_1



(b) F_2



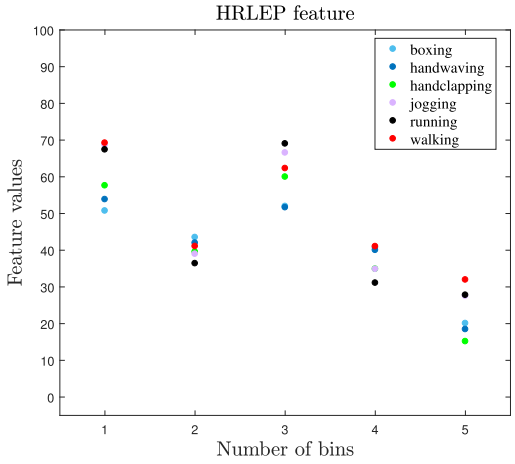
(c) F_3

FIGURE 3. Local features ($\mathcal{F}_L$) for six human actions in E-KTH dataset. (Values are normalized in the 0-100 region for visualization).

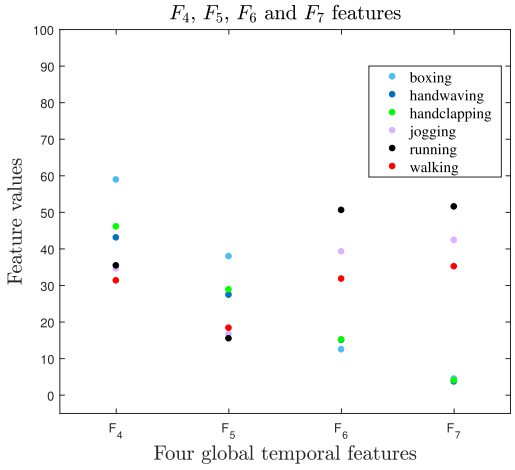
the discriminative features from $\mathbb{R}$. Then the global temporal feature vector, $\mathcal{F}_G(\mathbb{E})$ consisting of the 5-bin $\mathbb{H}$ and four other global features considering both $\mathbb{R}$ and $\mathbb{E}$ for the whole event stream as follows: $\mathcal{F}_G(\mathbb{E}) = \{\mathbb{H}, F_4, F_5, F_6, F_7\}$, where

$$F_4 = \max(\mathbb{R}), \quad (12)$$

$$F_5 = \max(W), \quad (13)$$

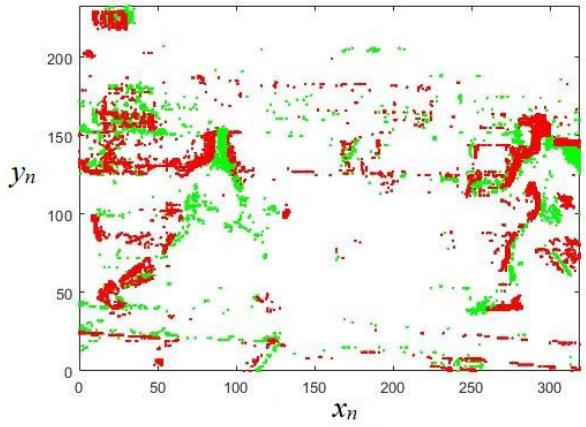


(a) HRELP ($\mathbb{H}$) features

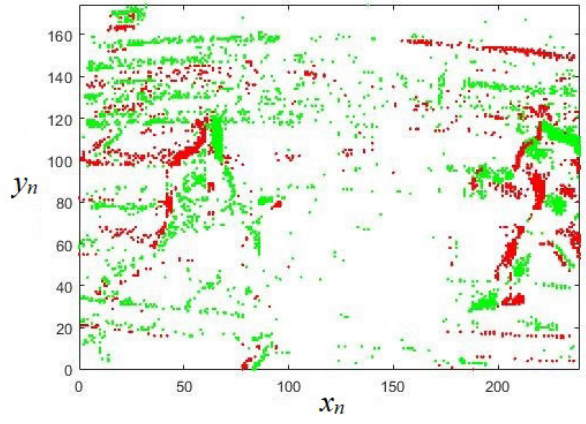


(b) F_4, F_5, F_6, F_7 features

FIGURE 4. Global features ($\mathcal{F}_G$) for six human actions in E-KTH dataset. (Values are normalized in the 0-100 region for visualization).



(a) Emulator-based extraction (E-UCF50)



(b) NVS device re-recording-based (R-UCF50)

FIGURE 5. Two examples for the same frame from a fencing sequence in UCF50 dataset explaining the amount and the distribution of the events in each frame: (a) PIX2NVS emulator has been used to generate the stream of the events and (b) The DVS240C camera has been used to acquire the events. For visualisation, the ON and OFF events are plotted with green and red colours, respectively.

F_6 and F_7 are the number of ON and OFF events in $\mathbb{E}$, respectively. The global features extracted from six sequences of the E-KTH dataset are shown in FIGURE 4 as an example.

E. FEATURE FUSION AND CLASSIFICATION

Finally, both $\mathcal{F}_L(\mathbb{E})$ and $\mathcal{F}_G(\mathbb{E})$ are fused to construct an overall feature vector, $\mathbb{F}(\mathbb{E})$, as

$$\mathbb{F}(\mathbb{E}) = \{\mathcal{F}_L(\mathbb{E}), \mathcal{F}_G(\mathbb{E})\}. \tag{14}$$

This $\mathbb{F}(\mathbb{E})$ is a 30 dimensions feature vector to represent the action in $\mathbb{E}$, and it is used to train the classifier for recognising the actions.

We conducted our experiments with several classifiers and found that the best results are obtained with KNN and QSVM. On one hand, from the complexity perspective, these classifiers have less complexity, especially KNN, compared to other classifiers. On the other hand, these classifiers are commonly used in the applications of computer vision for their efficiency, therefore, it is easy to compare with the existing work.

IV. PERFORMANCE EVALUATION

This section reports the extensive experiments conducted using challenging datasets to evaluate the performance of the proposed methodology for using NVS data for human action recognition.

A. DATASETS AND EXPERIMENTS SET UP

The publicly available and widely used NVS datasets can be categorised into three main groups: Emulator based datasets generated from the commonly used RGB datasets; datasets of NVS devices based re-recording of RGB video displayed on a monitor and datasets of actions acquired by native NVS devices. In our naming of datasets we identify these three groups with the prefixes E-, R- and N-, respectively in