Journal of Manufacturing Systems 68 (2023) 12–24

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

journal homepage: www.elsevier.com/locate/jmansys

Technical paper

Classification of primitive manufacturing tasks from filtered event data

Laura Duarte, Pedro Neto *

Centre for Mechanical Engineering, Materials and Processes (CEMMPRE), University of Coimbra, 3030-788, Coimbra, Portugal

ARTICLE INFO

Keywords:
Task classification
Manufacturing
Event data
Deep learning
Collaborative robotics

ABSTRACT

Collaborative robots are increasingly present in industry to support human activities. However, to make the human–robot collaborative process more effective, there are several challenges to be addressed. Collaborative robotic systems need to be aware of the human activities to (1) anticipate collaborative/assistive actions, (2) learn by demonstration, and (3) activate safety procedures in shared workspace. This study proposes an action classification system to recognize primitive assembly tasks from human motion events data captured by a Dynamic and Active-pixel Vision Sensor (DAVIS). Several filters are compared and combined to remove event data noise. Task patterns are classified from a continuous stream of event data using advanced deep learning and recurrent networks to classify spatial and temporal features. Experiments were conducted on a novel dataset, the dataset of manufacturing tasks (DMT22), featuring 5 classes of representative manufacturing primitives (PickUp, Place, Screw, Hold, Idle) from 5 participants. Results show that the proposed filters remove about 65% of all events (noise) per recording, conducting to a classification accuracy up to 99,37% for subjects that trained the system and 97.08% for new subjects. Data from a left-handed subject were successfully classified using only right-handed training data. These results are object independent.

1. Introduction

Collaborative robots play an increasingly important part in the manufacturing landscape. An efficient human–robot collaboration joins together the cognitive coordination, dexterity and flexibility abilities of humans with robot's accuracy and ease of execution of repeatable tasks [1,2]. In a shared workspace, collaborative robotic systems need to be aware of the human activities to (1) anticipate collaborative/assistive actions, (2) learn by demonstration, and (3) activate safety procedures. Robot situational awareness can be implemented at distinct levels, featuring the perception of elements in the current situation, comprehension of the current situation and the prediction of the element's future states [3]. This study tackles the first and second levels of the collaborative robot's situational awareness, i.e., the perception of elements and comprehension of the current situation, in the context of a set of primitive assembly tasks that together compose the complete assembly process of a given object. The recognition and prediction of human actions is key to make the robot aware of human actions in a shared workspace, promoting an effective and safe collaboration [4]. Reference studies rely on object detection and pose estimation from vision-based systems to infer human actions [5,6], as well as deep learning-based classifiers [7].

Vision is a sense humans use to perceive the world. As such, machine vision is often favoured in robotic systems to capture information about the environment. Traditionally, machine vision relies on the use of frame-based cameras, which benefit from years of continuous improvements in both equipment and machine vision algorithms. In the last few years, we have witnessed an increase in image resolution, which requires more storage memory and demands complex methods to achieve the classification of human actions. Additionally, frame-based cameras performance suffers from motion-blur, high latencies and low dynamic range. These issues promoted the development of the event camera, an alternative vision sensor inspired by biological vision.

In this study, we propose to capture human actions (primitive assembly tasks) using a Dynamic and Active-pixel Vision Sensor (DAVIS) event camera. It detects changes in brightness in the captured scene, by measuring logarithmic light intensity asynchronously and for each of the sensor's pixels [8], making it suitable to detect motion. As event cameras naturally suffer from random noise, several filters are combined to remove event data noise. The proposed event-based dataset contains five representative assembly primitives. Data were collected from different subjects and assembled objects. Advanced deep learning and recurrent network classifiers combine the classification of spatial and temporal actions. Fig. 1 shows an overview of the proposed framework. The main contributions of this study are listed as follows:

- An efficient method of combining multiple event filters, consistently targeting and removing noise events. The event filters'

* Corresponding author.
E-mail address: pedro.neto@dem.uc.pt (P. Neto).

https://doi.org/10.1016/j.jmsy.2023.03.001
Received 19 October 2022; Received in revised form 10 February 2023; Accepted 1 March 2023
Available online 9 March 2023

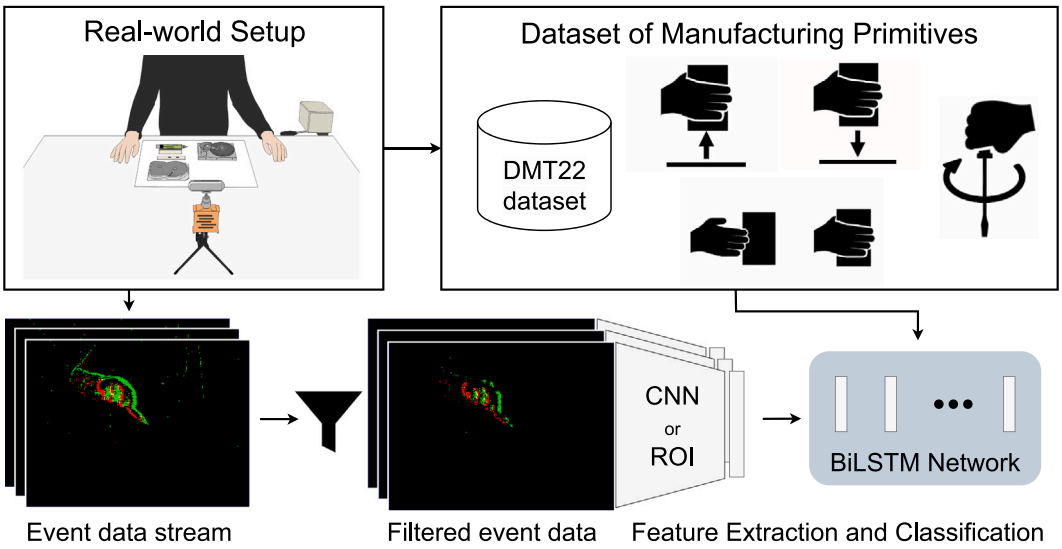

Fig. 1. The proposed framework to classify primitive manufacturing tasks from filtered event data. Assembly task patterns are classified from a continuous stream of event data through a convolutional neural network (CNN) or region of interest (ROI), after passing several filters to remove event data noise. The classifier, a Long Short-Term Memory (LSTM) network with a single bidirectional layer (BiLSTM) is trained on the proposed dataset of manufacturing tasks (DMT22).

performance is thoroughly tested on the metrics of data reduction and classification accuracy.

- A novel event-based dataset, the dataset of manufacturing tasks (DMT22), featuring five representative assembly primitive actions (PickUp, Place, Screw, Idle and Hold), collected from different subjects and assembled objects. Recorded data are available in an open-source dataset [9]. No event datasets focused on primitive manufacturing tasks are currently publicly available.
- Use of two distinct classification methods to evaluate the event filter's impact on classification accuracy. Different data selection methods further add variety. Comparison between the classification results obtained from a Recurrent Network (RN) architecture that takes as input the region of interest (ROI) features of the human hand captured by the event camera (RN-ROI) and from a deep learning architecture LRCN-TBR, combining a Long-term Recurrent Convolutional Network (LRCN) with the Temporal Binary Representation (TBR) method that converts the camera event output stream into image frames.
- Study of the impact of left-handed and right-handed subject data on classification accuracy.
- The proposed RN-ROI and LRCN-TBR classifiers are object independent due to the use of primitive actions;
- Evaluation of the proposed framework in the assembly of different objects by different subjects working in unstructured environment.

2. Related studies

2.1. Event camera

One of the most well-known event cameras, the Dynamic Vision Sensor (DVS) [10], operates by measuring temporal contrast, which is characterized by changes in brightness in the captured scene. The brightness of a scene is defined by the logarithmic light intensity, log(I). The DAVIS combines the DVS with an active pixel sensor (APS) at the pixel level. In each DVS pixel, the current log(I) is continuously compared to the memorized log(I) value of the same pixel. The pixel generates an asynchronous event when either a lower contrast threshold, θ_{OFF}, or an upper contrast threshold, θ_{ON}, is exceeded, Fig. 2. After firing the event, the pixel memorizes the current log(I) value and then resets itself in order to capture the next change in scene brightness. Each generated event, $\boldsymbol{e}_i = (x_i, y_i, ts_i, pol_i)$, is indexed using a unique temporal index i and contains the information about the event's coordinates in the pixel array, x and y, the timestamp of the occurrence of the event, ts, and the sign of the associated brightness change, known as the event's polarity, pol. For a positive brightness change an ON event (positive event) is created, $pol = 1$, and for a negative brightness change an OFF event (negative event) is created, $pol = 0$.

2.2. Event filters

2.2.1. Background activity filters

Event cameras naturally suffer from random noise, due to thermal noise and leakage currents from the event camera itself [11]. Background activity noise occurs when a pixel randomly outputs an event without being triggered by a brightness change in the scene.

The first attempt at an event-based Background Activity filter registered spatio-temporal neighbours in a timestamp map [12]. Each new event's timestamp is stored in all eight neighbouring pixels on the timestamp map, Fig. 3. Then, if the new event's current timestamp exceeds the value of the timestamp map at this event's location by less than the support time, dt, the event goes through the filter. Otherwise, it is discarded. The timestamp map in this method uses one memory cell per sensor pixel, $O(N^2)$.[1]

To reduce the amount of memory required, it was proposed in [11] to subsample the sensor's pixel array into groups of $s \times s$ pixels, with s as the subsampling rate. This reduces the amount of required memory to $O(N^2/s^2)$ cells. In this case, a new event's timestamp provides spatio-temporal support to all pixels inside its corresponding group within time dt. The main disadvantage of this method is that the spatio-temporal neighbourhood for each event is restricted to the events from the same group. Also, for the method to work properly, small values of s have to be used, such as $s = 2$ and $s = 4$, as shown in Fig. 4.

The Background Activity filter proposed in [13] challenges both previous attempts by only using a memory complexity of $O(N)$. This is accomplished by using two memory cells per row and two memory cells per column, which store all the necessary data for recovering recent events. By storing information in addition to the timestamp, the error is

[1] Space complexity expressed using the O notation, where N represents the camera's pixel array size. $O(N) = O(W) = O(H)$, with W and H as the DAVIS pixel array width and height, respectively.

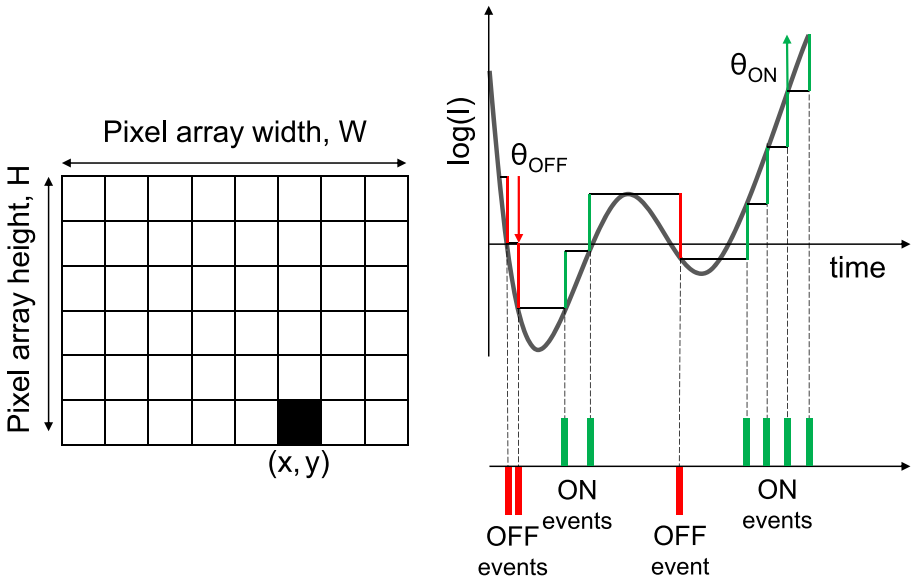

Fig. 2. Event generated in a DVS pixel. A pixel with coordinates (x, y) belonging to the DVS's $W \times H$ sensor array reads the captured logarithmic light intensity, log(I), over time and responds in the form of ON and OFF events when the upper or lower contrast thresholds, θ_{ON} and θ_{OFF} respectively, are exceeded.

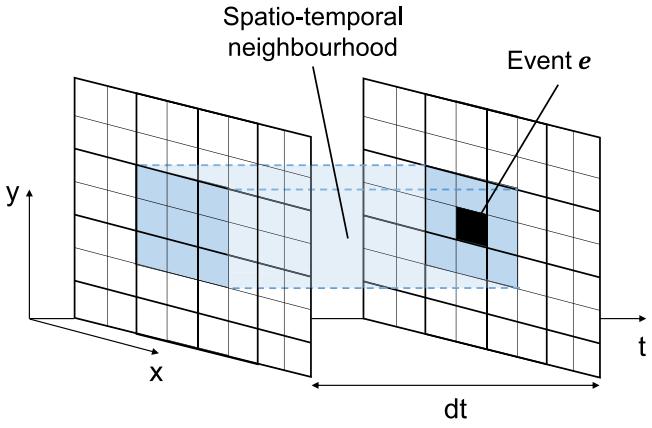

Fig. 3. Illustration of the spatio-temporal neighbourhood as in [12]. The event e will only pass through the Background Activity filter if another event exists within its spatio-temporal neighbourhood (blue). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

significantly reduced by allowing correlation with neighbouring events in future time.

Another study uses a hashing-based method [14], promoting a further reduction in storage costs. Additionally, this study also presents two metrics for the evaluation of the global performance of Background Activity filters, the percentage of correct predictions for real events and the percentage of real events in the filter's output. With the DAVIS240 rock–paper–scissors dataset, ROSHAMBO17 [15], Liu's filter [11] obtains state-of-the-art for these proposed metrics for neighbourhoods of duration $dt \geqslant 1.2$ ms and a $s = 2$ subsampling rate.

2.2.2. Customized filters

Some event filters must be applied on a case-by-case basis, e.g., to selectively reduce data flow and/or correct some innate issues with the event camera output. A frequent issue with event cameras is that complex circuit bias and manufacturing imperfections cause biased pixels and mismatch contrast threshold among pixels [16]. An example is the presence of hot pixels, which are pixels with low contrast thresholds that continuously fire events when the visual input is idle. There are two distinct solutions to this issue: (1) record with a stationary event camera and use jAER's *HotPixelFilter* to identify and then block the pixels with high event rates, and (2) implement an additional rule in one of the previously discussed Background Activity filters which prohibits self-correlation as a pixel's spatio-temporal support.

Event camera pixels naturally have a refractory period, defined as the duration in time that the pixel ignores brightness changes after an event is generated. The larger the refractory period the fewer events are produced by moving objects [17]. This type of filter has been used in [18] to follow the actual position, in time and space, of a helium-filled soap bubble (HFSB) rather than the trace left a few milliseconds after its passage. Alongside fast motion, this filter can also be applied to broad moving edges, along which contrast changes continuously [19]. In jAER's *RefractoryFilter* the concept is taken a step further, as it can either filter or pass events within the predefined refractory period. By allowing passage to all events that occur within the refractory period, this filter can work similarly to a self-correlation filter.

The transition between ON and OFF events can be exploited for tracking applications. For example, in [20], active LED markers have successfully been used for pose tracking of a quadrotor by analysing these transitions in the DVS data. In [18], HFSBs are used as flow tracers, by generating a pixel-sized localized response in the DVS, to reconstruct the 3D path and velocity inside a wind tunnel. To distinguish the events generated by HFSBs and noise, the "pair filter" is used to only pass events which consist of an ON/OFF pair. The jAER *OnOffProximityLineFilter* works similarly, by only outputting events that are supported by a nearby event (neighbour) of the opposite polarity.

The combination of multiple filters is often not fully explored nor detailed in literature. This paper identifies some of the most relevant filter algorithms and builds a novel framework featuring event filters with shared computational resources.

2.3. Action classification

Gestures are action primitives frequently used as a natural human–robot interface. A set of 25 gestures, used to interface robots, are recognized using two 3D Convolutional Neural Networks (CNNs) and a Long Short-Term Memory (LSTM) network in [21]. To teach assembly tasks, multiple gestures can be used sequentially and classified as either directional (Up, Down, Left, Right), orientational (90°, 180°), manipulation (Install, Remove, PickUp, Place), and feedback type (Confirm, Stop) [22]. Human action sequences can be segmented into primitive

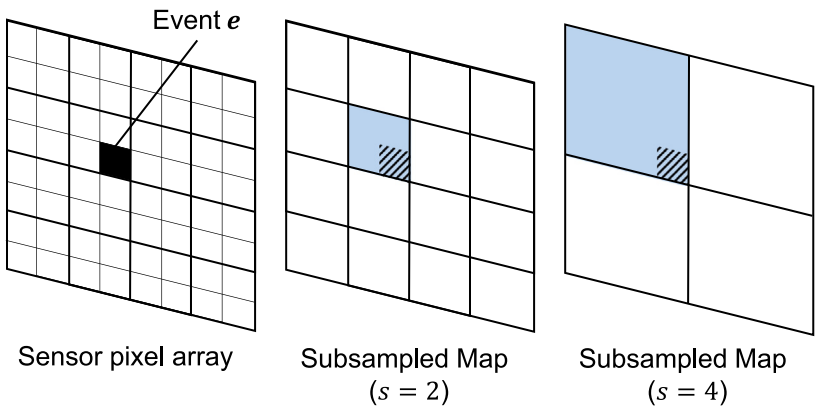

Fig. 4. Spatio-temporal support (blue) provided by an event on the sensor grid (left) when using a subsampling rate of $s = 2$ (middle) or $s = 4$ (right). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

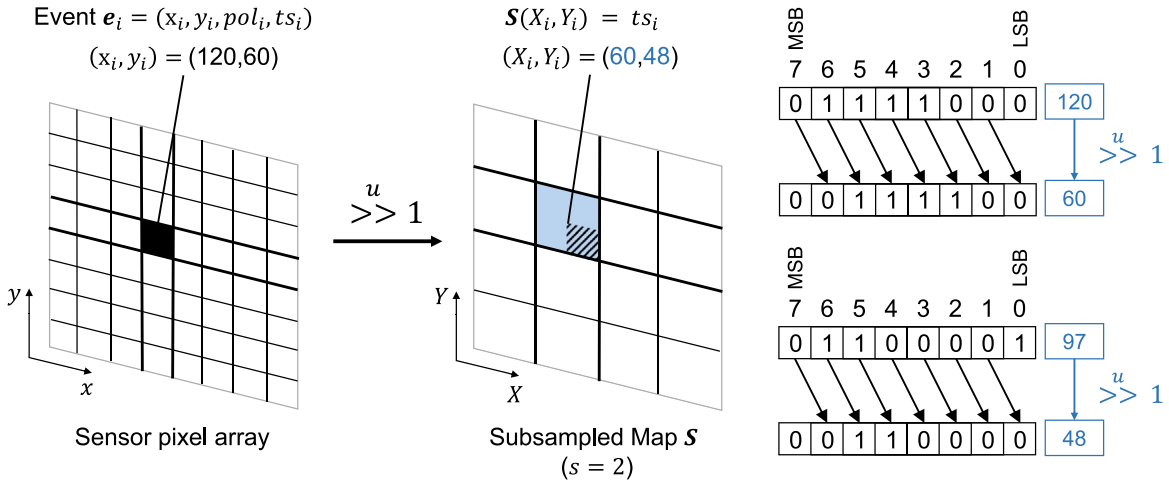

Fig. 5. A single right unsigned bit shift, $\overset{u}{\gg} 1$, is used to convert event's coordinates into subsampling map coordinates. MSB and LSB stand for the most significant bit and least significant bit, respectively.

Table 1
Event-based datasets for action and gesture recognition.

Author	Year	Dataset name	Event camera model	Content description
A. Amir [28]	2017	DvsGesture	DVS128	11 hand gestures under 3 lighting conditions
I. Lungu [15]	2017	ROSHAMBO17	DAVIS 240	3 Rock–Paper–Scissors hand gestures
S. Baby [29]	2018	DVS Gesture data	DVS128	10 hand gestures
E. Ceolini [30]	2020	DVS-EMG dataset	DVS128	5 sign language gestures. Additional APS frames and EMG data
S. Innocenti [31]	2020	MICC-Event Gesture dataset	Prophesee GEN3S VGA-CD	Extension of DvsGesture under more challenging conditions

tasks and then recognized by learning the relation between detected objects and human pose from RGB-D data through the use of graph networks [23]. Example action classes considered are Idle, Place, Hold, Pour (Kitchen context) and Screw (Workshop context). Human actions can also be divided into a set of static gestures (SGs) and dynamic gestures (DGs) [24]. An interesting application of segmenting human activity into a sequence of individual actions is presented in [25], where a robot can detect forgotten actions and then remind the human of such actions. Robots themselves can also benefit from using primitive tasks to adapt to build complex behaviours. These primitives are usually taught through human demonstration and can then be combined to perform product assembly [26,27]. There is a relatively small number of event-based datasets for action and gesture recognition, Table 1. None of the datasets available is focused on primitive manufacturing tasks. Primitive actions such as pick up, place, screw, etc. cover the most common manufacturing assembly tasks and can be used sequentially to perform complex assembly processes [4,6,7].

3. Methodology

3.1. Event filters

As previously mentioned, the use of subsampling as a basis for a Background Activity filter achieves good results. By subsampling the sensor's pixel array into groups of $s \times s$ pixels, a subsampled map is created, in which the most recent timestamp (from the last event in the corresponding subsampled group) is stored. A right unsigned (u) bit shift, $\overset{u}{\gg} n$, is a bitwise operation which shifts the operand by n bits to the right. In this operation, excess bits shifted off to the right are discarded and the vacant bit positions from the left are filled with zeros. Unsigned bit shifts are a efficient way to perform the division of unsigned integers by powers of two. A single right logical bit shift, $\overset{u}{\gg} 1$, can be used to directly map events from the sensor pixel array to the subsampled map corresponding to $s = 2$, Fig. 5. The only drawback of this operation is that the subsampling rate s must be a power of two,

$s : s = 2^n, n \in \mathbb{N}$. Assuming a new event, defined as $\boldsymbol{e}_i = (x_i, y_i, ts_i, pol_i)$, the coordinates x_i and y_i are shifted by n bits into the corresponding subsampled map coordinates, X_i and Y_i:

$$\begin{aligned} X_i &= x_i \overset{u}{\gg} n \\ Y_i &= y_i \overset{u}{\gg} n \end{aligned} \tag{1}$$

After each new event $\boldsymbol{e}_i$ passes through all the filters, or is caught by any one of the filters, its timestamp ts_i is stored into the subsampled map $\boldsymbol{S}$:

$$\boldsymbol{S}(X_i, Y_i) = ts_i \tag{2}$$

To avoid spatio-temporal correlation with non-events, the following condition, called the First Event filter, prohibits correlation with the zero-initialized subsampled map $\boldsymbol{S}$, by only passing events if:

$$\boldsymbol{S}(X_i, Y_i) > 0 \tag{3}$$

Since most initial events from a recording are noise, the First Event filter substantially reduces the initial spike of noise events despite its finite filtering capacity. In the Background Activity filter, the value of the subsampling map $\boldsymbol{S}(X_i, Y_i)$ is compared to the current event's timestamp, ts_i, to evaluate if their difference is less than the pre-defined support time dt_{BA}. The new event $\boldsymbol{e}_i$ is filtered if:

$$ts_i - \boldsymbol{S}(X_i, Y_i) \geqslant dt_{BA} \tag{4}$$

When using a Background Activity filter, a convenient process to filter hot pixels is to negate self-correlation. This means that if the only spatio-temporal neighbour for a new event is an older event at the same pixel coordinates, the new event is deemed invalid. However, spatial information of the events is lost when subsampling occurs, because only the timestamp is stored at the subsampling map $\boldsymbol{S}$. As such, to build a Hot Pixel filter, a new map called the coordinate map $\boldsymbol{C}$, with the same dimension as $\boldsymbol{S}$, will store the coordinates of the last event, x_i and y_i, at $\boldsymbol{C}(X_i, Y_i, 0)$ and $\boldsymbol{C}(X_i, Y_i, 1)$, respectively. Thus, an event will only pass the Hot Pixel filter if:

$$\boldsymbol{C}(X_i, Y_i, 0) \neq x_i \wedge \boldsymbol{C}(X_i, Y_i, 1) \neq y_i \tag{5}$$

After each new event $\boldsymbol{e}_i$ is either caught by or passed through any of the event filters, its x and y coordinates are stored in the coordinate map $\boldsymbol{C}$:

$$\begin{aligned} \boldsymbol{C}(X_i, Y_i, 0) &= x_i \\ \boldsymbol{C}(X_i, Y_i, 1) &= y_i \end{aligned} \tag{6}$$

A similar approach to the Background Activity filter is used to build the Refractory filter, but instead, it uses a smaller support time, dt_{Refr} ($dt_{Refr} < dt_{BA}$), and filtering $\boldsymbol{e}_i$ if:

$$ts_i - \boldsymbol{S}(X_i, Y_i) \leqslant dt_{Refr} \tag{7}$$

If this condition is met, this means that an event is most likely noise related to flickering. Assuming that both general background activity and hot pixels have been filtered out, the only non-essential information left on the image is the shadow cast (hand and object in this study). These events are not previously filtered due to the fact they create spatio-temporal neighbourhoods which pass through the Background Activity filter. These events have, generally, the same polarity. By checking neighbouring subsample groups for spatio-temporal support from events of the opposite polarity, it is possible to determine if events are part of a shadow. Taking inspiration from the Background Activity filter method, the Polarity filter will use a similar search method, but in a wider spatio-temporal neighbourhood. A range of coordinates is used instead of a single comparison, and a greater support time, dt_{Pol} ($dt_{Pol} > dt_{BA}$), is used. However, only neighbouring events with opposite polarity, $1 - pol_i$, are used to validate the condition. The event passes the filter if, for any possible coordinate combination within range, the following condition is true:

$$\begin{aligned} &ts_i - \boldsymbol{S}(range_x, range_y, 1 - pol_i) \leqslant dt_{Pol}, \\ &\forall range_x \in \{X_i - 1, X_i, X_i + 1\}, \forall range_y \in \{Y_i - 1, Y_i, Y_i + 1\} \end{aligned} \tag{8}$$

where $\boldsymbol{S}(X_i, Y_i, 1)$ and $\boldsymbol{S}(X_i, Y_i, 0)$ are the subsampled maps for positive and negative events, respectively. However, the reduced filtering capacity this filter offers is outweighed by its main issue of requiring the previous filter's conditions to check for both polarities. The polarity filter mostly serves the purpose of filtering shadows. However, since shadows are not prominently featured in the proposed dataset, this filter will not be used in this study.

3.2. Dataset of Manufacturing Tasks (DMT22)

The DMT22 dataset contains recorded data from an event camera, a depth camera and a magnetic tracking system. Recorded data are available in an open-source dataset [9]. While the magnetic tracker is attached to the human body, the cameras are stationary. In this study, we focus on data from the event camera. The dataset includes data from other sensors so that it can be used by a broader audience and results from different sensor data can be compared.

Focusing on the event camera, if it is moving, information about both moving and stationary elements can be obtained, due to the relative movement between them. This is the Eye-in-Hand system, where a camera can be held by a robot with pre-programmed movements to track the camera's exact pose in 3D space, Fig. 6 (left). With this, a scene can be viewed from different angles, obtaining information which can be used to reconstruct it in the world coordinate system, as witnessed frequently in Simultaneous Localization and Mapping (SLAM) applications. An alternative is to use an Eye-to-Hand system, in which the camera is placed at a fixed point in the workspace, Fig. 6 (middle). It facilitates the computation of transformations between image and world coordinate systems through calibration. When the camera is fixed, stationary elements will not be registered in events data, only the elements (human hand) moving relative to the camera will be captured. Thus, the Eye-to-Hand configuration is the more suitable configuration for capturing the manufacturing primitive actions in this dataset, Fig. 6 (right). The setup used for the acquisition of the dataset data is in Fig. 7.

The dataset includes a set of manufacturing assembly tasks, each featuring a different object, a Wii remote, a hard disk drive, and an electric screwdriver, Fig. 8. The objects were selected to be of equivalent complexity. All possess similar primitive tasks, including both "screwing" and "fitting" actions, which are key primitives in assembly tasks. Electro-mechanical items were chosen as they provide a balance between being a complex assembly case and being big enough that its components will be visible on the recordings.

The tasks were recorded by five subjects: 1 right-handed female subject (Subject 1) and 4 male subjects, of which 3 are right-handed (Subjects 2, 3 and 4) and 1 is left-handed (Subject 5). The selected subjects vary in gender, assembly experience and feature different dominant hands. All subjects were given a brief introduction to each assembly task before recording. Each subject performed all three assembly tasks four times. An exception was created for the left-handed subject, Subject 5, which performed every task both with a right-sided (right-handed) and left-sided (left-handed) placement of the parts to be assembled and the required tools. The right-sided setup was used for all right-handed subjects. The left-sided setup is a mirrored version of the right-sided setup, which means most parts and tools are on the left side of the subject, a more convenient placement for a left-handed subject. These two setups allow to study how well a left-handed subject action can be classified in a context where the classifiers are trained with right-handed subject's data. The dataset features a total of 72 recordings: 4 right-handed subjects × 3 tasks × 4 performances + 1 left-handed subject × 2 configurations × 3 tasks × 4 performances. Each

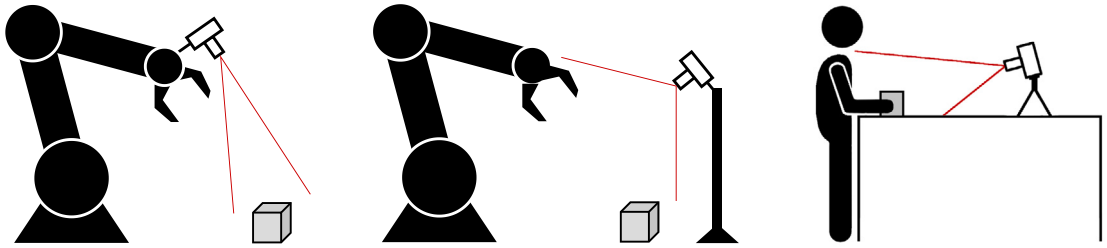

Fig. 6. Eye-in-Hand system configuration (left), Eye-to-Hand system configuration (middle), and the proposed camera setup (right).

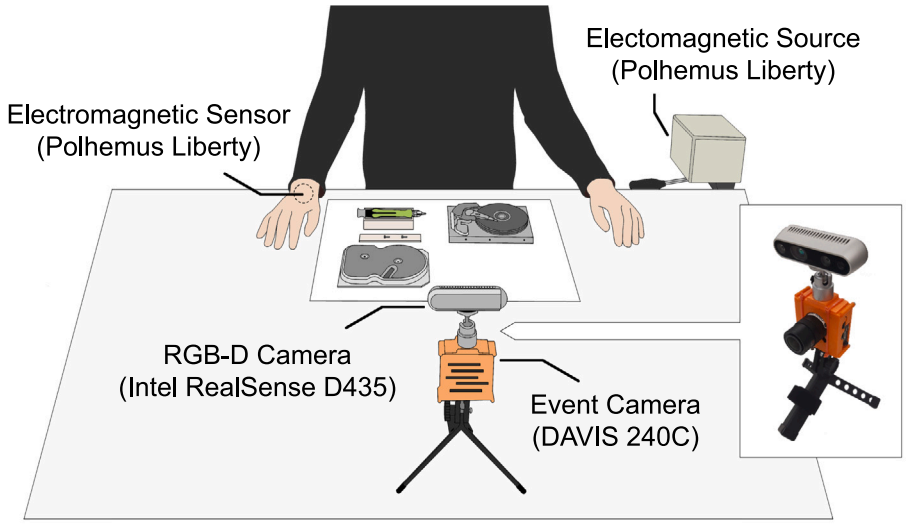

Fig. 7. The DMT22 dataset acquisition setup featuring the hard disk drive assembly task. The DAVIS event camera and the RGB-D camera are coupled to capture scene data. The electromagnetic sensor is attached to the subject's wrist.

task recording features multiple primitive tasks and lasts approximately 25 s.

The data were manually labelled by a single annotator, as to minimize attribution discrepancies. Action classes were labelled as one of the following five primitives: PickUp, Place, Screw, Idle and Hold. Fig. 8 shows a detailed description of the assembly of the Wii remote consisting of 13 different primitive tasks. The event camera used to record the dataset is a DAVIS 240C, capturing events with a resolution of 240 pixels × 180 pixels. DAVIS APS data are captured at 30 fps (frames per second), along with RGB-D data from the Intel RealSense Depth Camera D435 at 30 fps. The Polhemus Liberty electromagnetic tracker was used to capture pose data, at 20 Hz, from a sensor attached to the subject's wrist. Such data are expected to suffer from magnetic distortion.

3.3. Classification methods

Two classification methods are proposed to classify the primitive manufacturing/assembly tasks from filtered data:

1. RN-ROI: A recurrent network architecture that takes as input features from the region of interest (ROI) of the human hand.
2. LRCN-TBR: A deep learning architecture combines a Long-term Recurrent Convolutional Network (LRCN) with the Temporal Binary Representation (TBR) method [31] that converts the camera event output stream into image frames.

The RN-ROI method relies on a lightweight algorithm to define the ROI features of the human hand through object edge event activity [32]. The algorithm detects the pixel columns and rows featuring event activity to define a square ROI containing the human hand. The relevant features from the ROI are its centre coordinates, size, and percentage of active pixels. In [32], features are obtained at every 3000 events. In this study, a comparison will be made between the filtered and unfiltered data from the DMT22 dataset. As such, features must be obtained at fixed intervals of time for a fair comparison, so the same "frame" is represented in both analyses. The chosen interval of time to obtain features is $\Delta t = 30$ ms. Since the ROI features are presented sequentially and the intent is to use that temporal continuity to classify the primitive actions, a Long Short-Term Memory (LSTM) network with a single bidirectional layer (BiLSTM) is used. Features are zero-centred before classification. The schematic of the network is illustrated in Fig. 9. Since the DMT22 dataset is relatively small, a data augmentation technique is used to better train the LSTM network. All data are used in their original state as well as their mirrored state (horizontally flipped), effectively doubling the data available for training the network. This was also done to help the left-sided setup of Subject 5 to better conform to the rest of the DMT22 data.

For the LRCN-TBR method, the camera event output stream is converted into image frames. Given an interval of time Δt, a binary representation b^i is built for each pixel. For each possible event coordinate (x, y), the value of the representation is $b^i_{x,y} = 1$ in case any event is present and $b^i_{x,y} = 0$ otherwise. By considering a sequence of eight consecutive binary representations, these can be combined into a single 8-bit array, which is converted into a decimal number. The range of possible values corresponds to the greyscale range of 0−255. As such, this method is an efficient way to create images from events. The images created through this method cannot immediately be processed by a conventional LSTM network. Using Convolutional Neural Network (CNN) AlexNet, which is already trained, and by removing its last layers responsible for classification, the CNN is used to obtain features instead

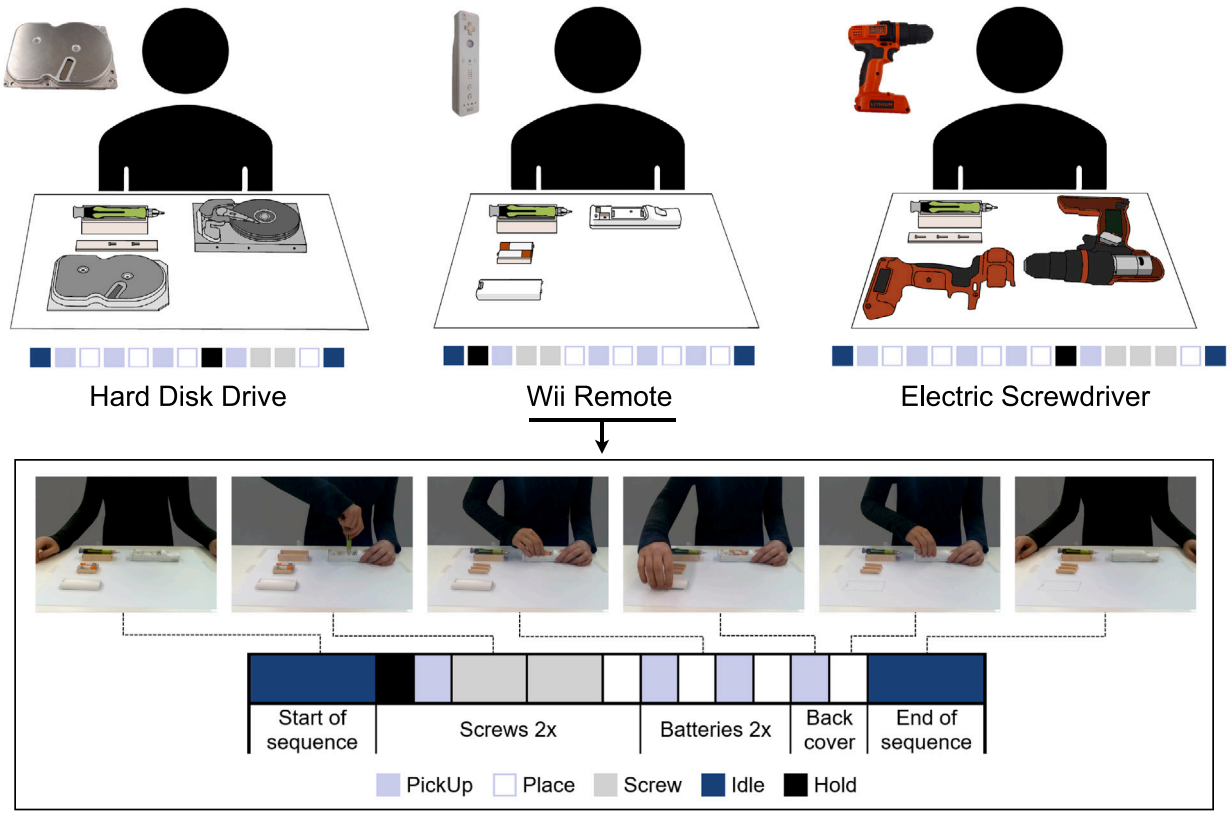

Fig. 8. The three different objects on the DMT22 dataset and the corresponding sequence of primitive tasks to complete the assembly of each one. Segmentation of a Wii remote assembly task from the DMT22 dataset, with each colour depicting a unique primitive action. It is composed by 13 different primitive tasks: (1) the human is idle, (2) holds the screwdriver, (3) picks up the screwdriver, (4) performs a screw action, (5) performs another screw action, (6) places the screwdriver, (7) picks up a battery, (8) places the battery, (9) picks up another battery, (10) places the battery, (11) picks up the back cover, (12) places the back cover, and (13) ends the assembly idle. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

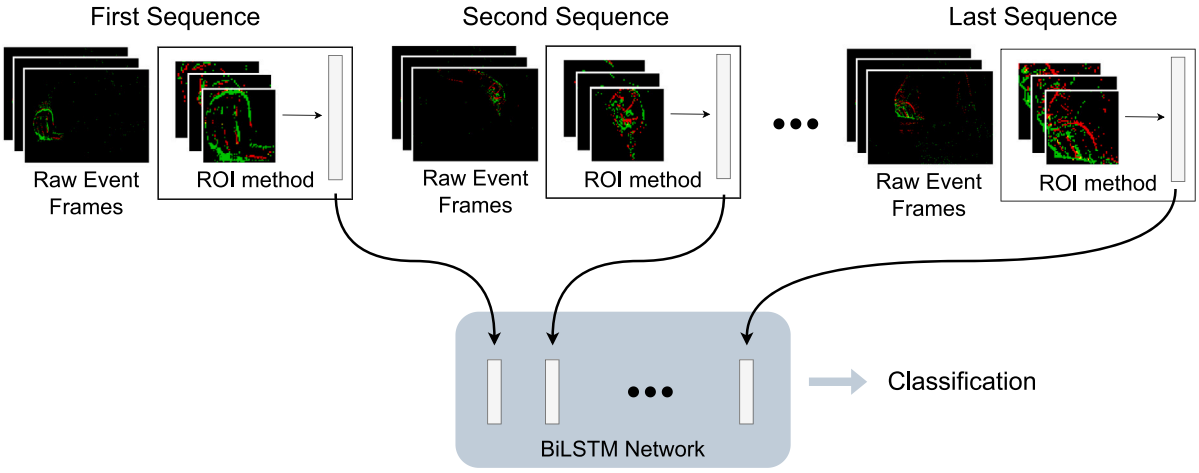

Fig. 9. Classification architecture of event data using the RN-ROI method.

of performing classification. The input frames must comply with the expected input size of the chosen CNN, AlexNet. The images created must first be cropped horizontally to a size of 227 × 180, to which vertical zero-padding up to 227 × 227 is then applied. As a final step, a LSTM network, identical to the one proposed for the RN-ROI method, can then obtain the temporal correlation from the analysed frames to classify the data. The schematic of the LRCN-TBR method is shown in Fig. 10.

4. Experiments and results

Detailed information on the experimental setup and data collected is presented in Section 3.2. The assembled objects are described, as

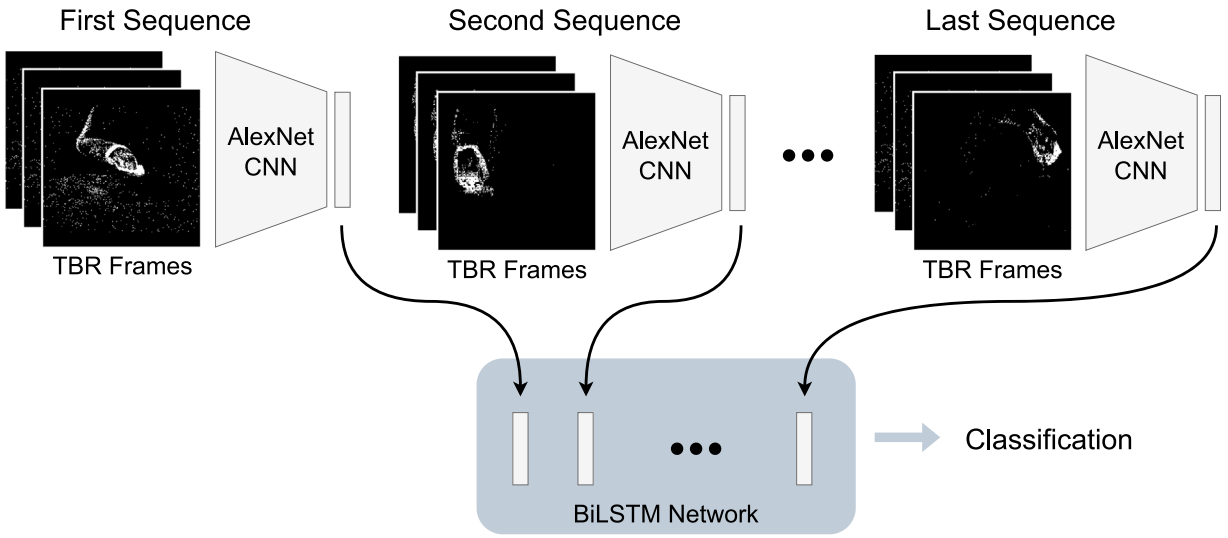

Fig. 10. Classification architecture of event data using the LRCN-TBR method.

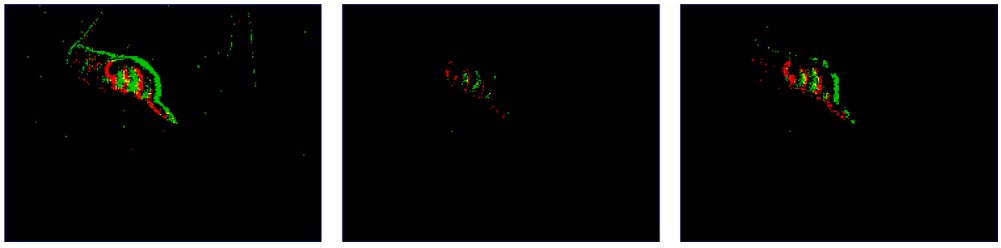

Fig. 11. Loss of information depending on subsampling rate. Original with no subsampling (left), with subsampling rate of $s = 2$ (middle) or with subsampling rate of $s = 4$ (right).

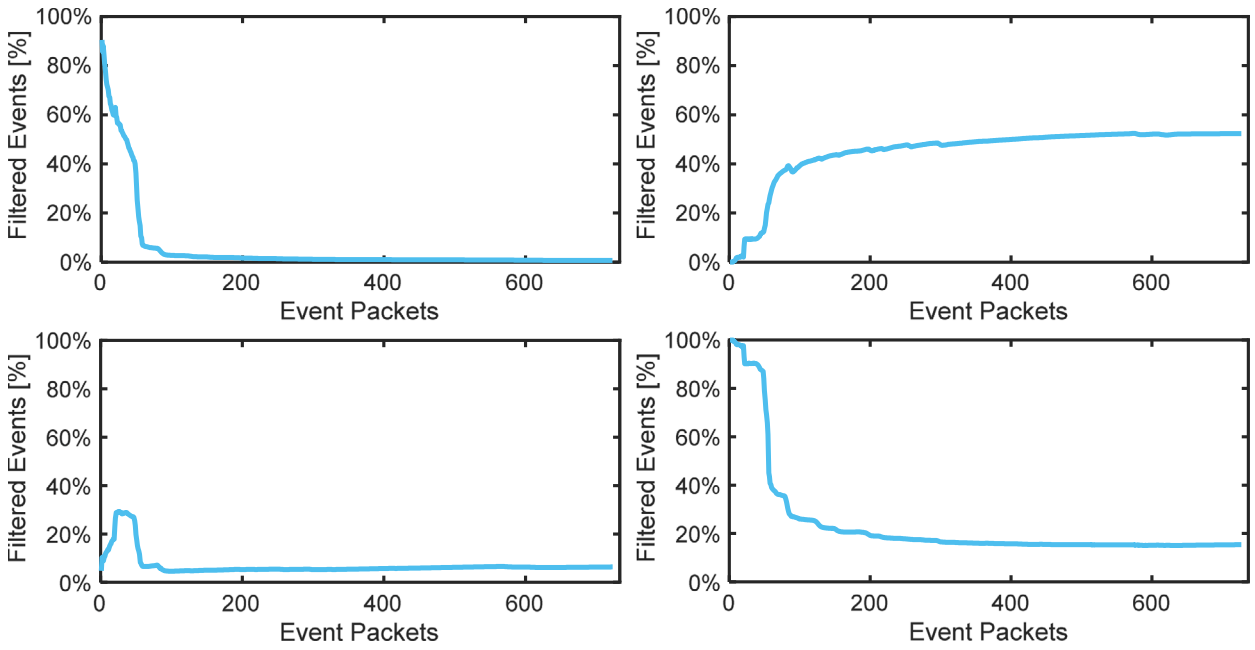

Fig. 12. Cumulative performance of individual filters (DMT22 dataset). First Event filter (top left), Background Activity filter (top right), Hot Pixel filter (bottom left) and Refractory filter (bottom right).

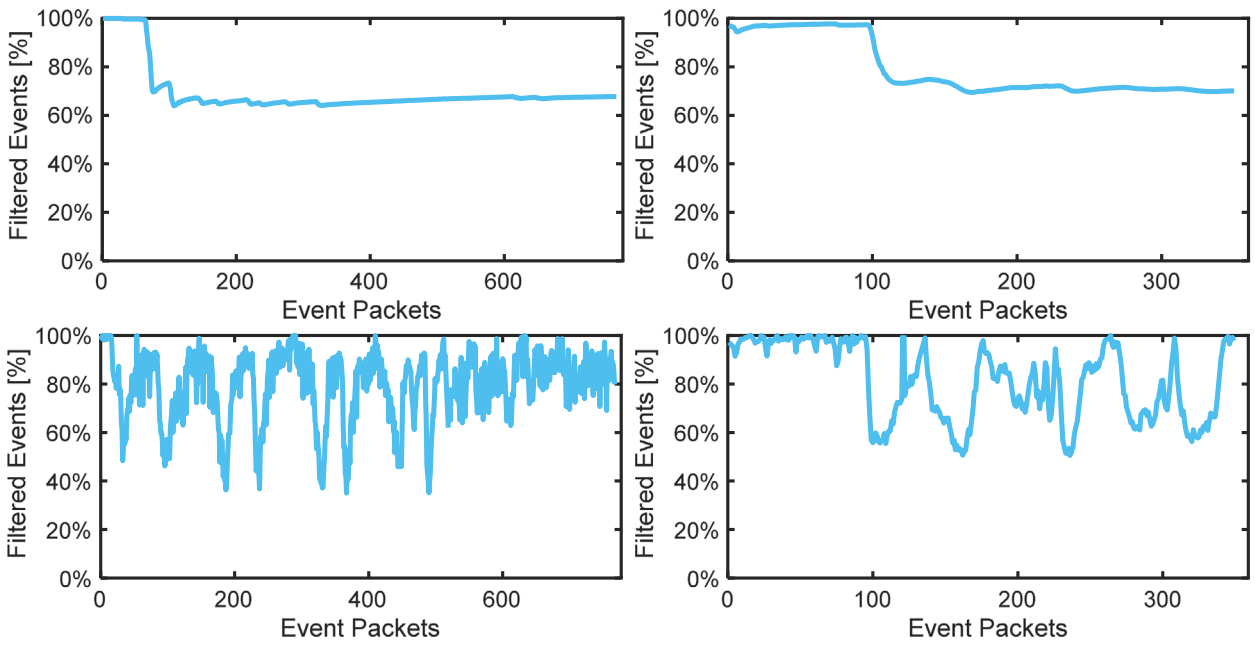

Fig. 13. Cumulative (top) and instantaneous (bottom) analysis of combined event filters on the DMT22 dataset (left) and on the EDHT21 dataset (right).

Table 2
Average results of event filters on Subject 1 data from the DMT22 dataset.

Object	Avg. % of filtered events for $s = 2$	Avg. % of filtered events for $s = 4$	Avg. number of events in sequence
Hard disk drive	85.89	67.39	661 874
Wii remote	82.58	63.59	478 685
Electric screwdriver	82.27	62.15	890 886

well as the subjects, experimental trials, data formats and equipment. The video that accompanies this article shows sample data collected for each assembled object.

4.1. Filter performance

The filtering results were obtained by applying them to the DMT22 dataset and the Event Data for Hand Tracking EDHT21 dataset [33]. The filters were built and tested in the open-source Java software framework jAER.[2] For each action sequence of the dataset, the event filters performance is measured by comparing the number of total events received to the number of filtered events.

$$\%\ \text{of Filtered Events} = \frac{\#\ \text{of events filtered}}{\#\ \text{of events in sequence}} \tag{9}$$

Filters are applied to event data sequentially, considering the parameters $dt_{BA} = 1.5$ ms and $dt_{Refr} = 10$ µs for the Background Activity filter and Refractory filter, respectively. Results in Table 2 were obtained from the DMT22 dataset for two distinct subsampling rates, $s = 2$ and $s = 4$. Subsampling rates have a high impact on filter performance.

Although the results have shown a higher filtering rate with a subsampling rate of $s = 2$, relevant features present in the event data are lost, Fig. 11. These features are essential for the recognition and classification of data-based action patterns. Filtering with $s = 4$ removes noise while leaving nearly all relevant events intact, creating a balance between noisy data and relevant features data. Further increasing the subsampling rate is not recommended, as it would considerably reduce

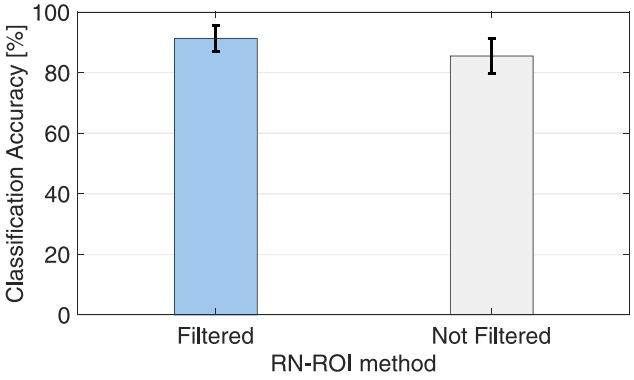

Fig. 14. Classification accuracy, with and without filters, applied on all the DMT22 event data, except left-sided object placement.

the filter performance. The event filter with a subsampling rate of $s = 4$ filters, on average, 60.76% of all events (average value of all subjects and object scenarios).

Considering an assembly sequence from the DMT22 dataset, the performance of each filter was evaluated by, at each event packet (collection of events grouped for processing purposes), comparing the number of total events received to the number of filtered events. By measuring these values cumulatively, at each new event packet, a graph is plotted showing the filter's performance along the sequence, Fig. 12, showing its effectiveness over time. It is important to note that, due to measuring cumulatively, most graphs will show fewer fluctuations in value for the percentage of filtered events when near or at the end of the sequences. The results for each filter for the first sequence of the DMT22 dataset (first hard disk drive assembly by Subject 1) are presented in Fig. 12.

Both the First Event filter, Fig. 12 (top left), and the Refractory filter, Fig. 12 (bottom right), behave similarly to an exponential decay function, in the sense that they filter almost all initial events of the sequence, but quickly lessen their filtering performance. They are effective to filter most initial noise events, which, due to their abundance, create spatio-temporal neighbourhoods which would pass through the other filters. The Hot Pixel filter, Fig. 12 (bottom left), also filters more

[2] Available at: http://jaerproject.net.

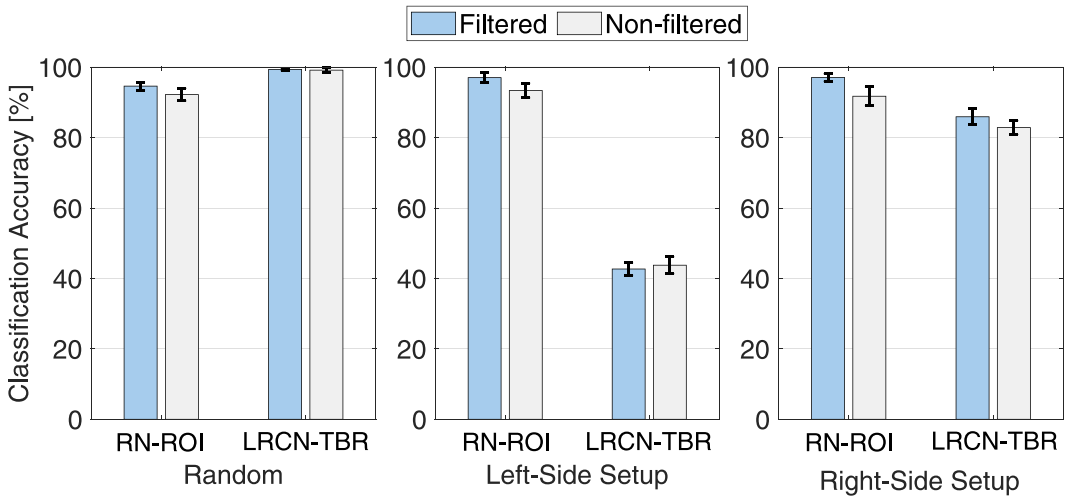

Fig. 15. Comparison of classification accuracy results from RN-ROI and LRCN-TBR methods, with and without filtered data. Three distinct scenarios were considered: randomly selected data (left), data from Subject 5's left-side setup (middle) and data from Subject 5's right-side setup (right).

events at the beginning of the action sequence, due to more noise which has similar behaviour to hot pixels. After that, the filter stabilizes the percentage of filtered events, indicative of actual hot pixels being consistently recognized and filtered. The Background Activity filter, Fig. 12 (top right), filters the most events out of all the individual filters presented. It filters consistently along the whole action sequence, filtering an average of 55% of incoming events. The initial build-up is related to the filter requiring past events to verify spatio-temporal correlation and these events still need to be accumulated.

The progression of the cumulative percentage of filtered events along time, Fig. 13 (top), mirrors the subject's behaviour in the recorded sequence. At the beginning of the sequence, the subject is idle. During this period, all events that are output by the camera are noise and, ideally, the filters should act to filter them. At the moment the subject starts moving, an abrupt decrease in the percentage of filtered events can be observed. This decrease of about 35% is relative to the events created when the human subject moves the hand and object during the action. These events provide important information and should, in fact, not be filtered. Except for slight fluctuations, the percentage of filtered events during the actual sequence is mostly constant, due to the events created by noise being produced at a steady rate, and the hand moving at mostly the same speed and a consistent distance from the camera. The behaviour of combined filters during the first 3D sequence from the EDHT21 dataset, Fig. 13 (top right), is very similar to that of the DMT22 dataset, Fig. 13 (top left).

To better evaluate filters performance, a graph is plotted with instantaneous values of filter performance, Fig. 13 (bottom). At each event packet, the total number of events received from the packet is compared to the number of events filtered from that same packet. The main difference between using the cumulative and instantaneous methods of plotting is that the latter shows a more descriptive representation of the filter performance during each assembly task.

When the subject hand is moving, the percentage of filtered events decreases significantly. During the transition between different motion directions, the brief pause of movement of the subject can be identified in Fig. 13 (bottom left) by the spikes at values of 100% filtered events. In the EDHT21 dataset it is possible to identify specific changes in movement direction from the instantaneous behaviour of the event filter. In the sequence in Fig. 13 (bottom right), the subject changes the direction of movement six times, which translates into a graph with six spikes at values of 100% filtered events, although the exact location of each spike is not clearly defined. This behaviour, if consistently identified, can be used as a feature for action segmentation and, consequently, improve classification accuracy.

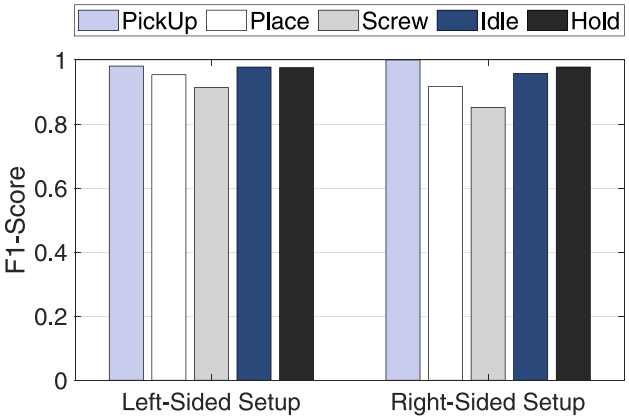

Fig. 16. F1-score per class from the DMT22 dataset using the RN-ROI method.

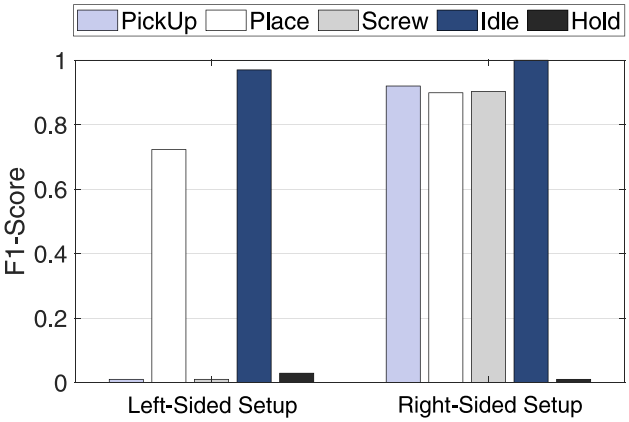

Fig. 17. F1-score per class from the DMT22 dataset using the LRCN-TBR method.

4.2. Classification results

The filters performance was evaluated by measuring the general classification accuracy by feeding the RN-ROI with filtered and non-filtered data. The LSTM network uses data from all five subjects in a division of 80% training, corresponding to the data of 4 subjects, and 20% testing, corresponding to the data of a single subject. In this

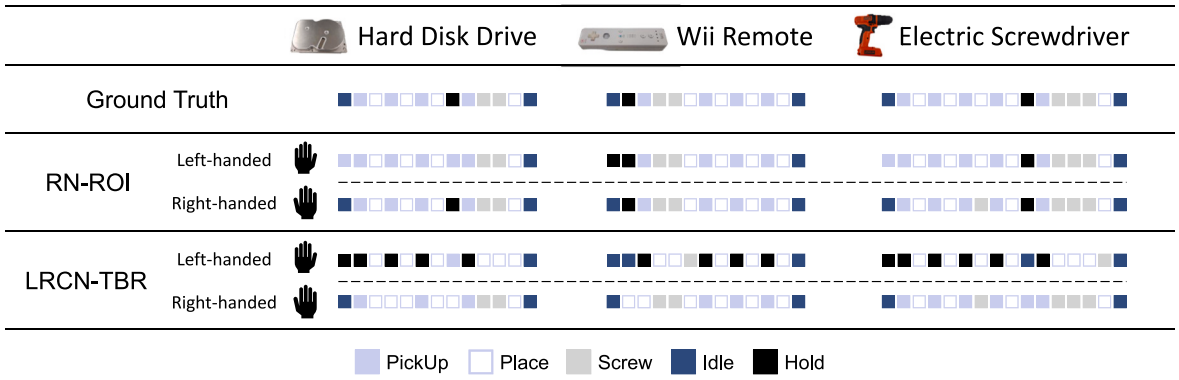

Fig. 18. Primitive assembly tasks classified in a continuous stream of data from the DMT22 dataset using the RN-ROI and LRCN-TBR methods. Results show two sequence lines for RN-ROI and LNCR-TBR, representing the classified task primitives when the subject testing the system is left-handed (top line) and right-handed (bottom line). Comparing the results with the ground truth primitive tasks, it is noted that the classification results obtained by the right-handed subject are better than the ones obtained by the left-handed subject, especially when using the LRCN-TBR method. This is likely because the DMT22 dataset is mostly composed by data collected from right-handed subjects.

particular test, only Subject 5's right-hand object placement is used. By isolating a specific subject's data for testing, such data are uncorrelated to the training data and the network has no previous knowledge of this particular subject. The results obtained through this method are expected to be worse than those obtained through a random selection of data, but they are much more meaningful due to resembling a real-world scenario, where a new subject is introduced to the system. The subject that was chosen for testing is alternated between all possible combinations, as cross-validation, to test both the data and the network to their fullest. The general classification accuracies (average and the standard deviation) are in Fig. 14. They show that the use of event filters improves classification. The gain in classification accuracy is, approximately, 6 percentage points, increasing from an average accuracy of 85.49% to an average accuracy of 91.28%. The standard deviation error is reduced when using the filters. For all network runs, the filtered result accuracy always surpassed the non-filtered result.

A second set of tests was conducted to observe the classification accuracy in the context of different data selection scenarios:

1. Random sample selection featuring 85% training data and 15% testing data from the dataset. This is a common classification accuracy metric, where testing data are not available as training data, but the system is trained with data from Subject 5, which might bias the results.
2. Subject 5's left-sided setup uses training data from all subjects, except Subject 5, and uses testing data from Subject 5's left-sided setup (Subject 5's left-hand object placement). The classifier has no previous knowledge of Subject 5.
3. Subject 5's right-sided setup uses training data from all subjects, except Subject 5, and uses testing data from Subject 5's right-sided setup (Subject 5's right-hand object placement). The classifier also has no previous knowledge of Subject 5.

For each of the scenarios above, the RN-ROI and LRCN-TBR classification methods were evaluated. The results in Fig. 15 show that, for almost all cases, accuracy improves by using the event filters. Also, the variability of results, characterized through the standard deviation error, reduces when using filtered data. The overall best classification results are obtained for the filtered random selection scenario, with the LRCN-TBR's $99,37\%$ accuracy outperforming the $94,63\%$ accuracy from RN-ROI. The classification accuracies for Subject 5's recordings are very high when using the RN-ROI classifier, proving that the system is reliable to be used by right-handed and left-handed subjects, even after training the system without data from left-handed subjects. For the RN-ROI approach, the classification results of Subject 5's left-sided and right-sided setup recordings are identical, with accuracies of $97,03\%$ and $97,08\%$, respectively. This suggests that, when using the mirror data augmentation technique, it is favourable to accommodate the left-handed subject with a left-sided configuration of objects, without loss of classification accuracy. For the scenarios of the left-side setup and right-side setup, the classification accuracy significantly degrades when using LRCN-TBR method, especially for the left-side setup. In such a context, it can be concluded that deep learning based LRCN-TBR method is more dependent on the training data from a specific subject than RN-ROI.

The F1-score is chosen as an auxiliary metric to evaluate classification performance due to the imbalance in the class distribution of the DMT22 dataset. The F1-score evaluates precision and recall, comparing the number of correct guesses (TP) against the number of other tasks which are misclassified as the intended class (FP) or against the number of intended tasks which are misclassified as another task (FN). The F1-score is calculated for each class from the DMT22 dataset, demonstrating similar accuracy when using RN-ROI, Fig. 16.

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \tag{10}$$

$$precision = \frac{TP}{TP + FP} \tag{11}$$

$$recall = \frac{TP}{TP + FN} \tag{12}$$

The LRCN-TBR method falls short of classifying data from new subjects (which are not represented in the training dataset), especially in the case of the left-sided setup where classification accuracy reached at most $43,77\%$ using non-filtered data. One reason for this behaviour might be that CNN-based features cannot characterize movement, while tracking-based features, such as ROI features do. However, this does not explain why the left-sided and right-sided setups have very distinct classification accuracies. To further analyse this incongruency, the F1-score was calculated, per class, to compare the two setups for the LRCN-TBR method, Fig. 17. The Idle class is the only class that is always correctly classified and both setups struggle to classify the Hold task. This makes sense, as the Idle class has the most distinct frames out of all the tasks (almost no events). Also, the Hold task is the most difficult task to classify as it has the least data for the network to learn and features a lot of similarities to the Place task. The Place task is well classified in both setups. The left-sided setup struggles most in the PickUp and Screw tasks. The misclassifications in these cases are not consistent, which indicates that the network does not have a good grasp on the features which characterize these tasks.

Fig. 18 shows task predictions for each object and method against the ground truth. Following previously discussed results, the Hold task is the most frequently misclassified.

5. Conclusion and future work

A novel methodology to classify manufacturing assembly primitives from event data was presented. Results demonstrated the effectiveness of the proposed deep learning and recurrent network classifiers, especially when event data are filtered. The combination of different filters promoted a dynamic response of the system, consistently targeting and removing noise events. On average, they filter out 65% of the events from each recording. The classification accuracy, evaluated on the proposed DMT22 dataset, is about $91,31\%$ (using filters), 6 percentage points higher than the classification accuracy obtained without using filters. In such a context, it can be concluded that the multiple filters play a key role in the classification accuracy when using event data as main input source. In addition, less data makes the classification faster and require less storage resources. It can also be concluded that in general the RN-ROI method presents better classification accuracy than the deep learning based LRCN-TBR method, especially when used by left-handed subjects who did not train the system. LRCN-TBR is more dependent on the training data from a specific subject than RN-ROI.

From a practical application perspective, the classified primitives can serve as input for a human–robot collaborative system that anticipates the co-worker's needs (bringing parts and tools to the assembly workplace), learns from the co-worker's demonstrations, and activates safety procedures according to the actual tasks being performed by the human co-worker. The proposed methodology has the added benefit of being object independent.

The proposed DMT22 dataset of manufacturing primitives should be extended and complemented with more data (novel assembly scenarios and new subjects). Movements that are not defined as actions, such as transition movement between tasks, should also be considered and labelled, aiming to obtain better classification and hand tracking results.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported by Fundação para a Ciencia e a Tecnologia [2021.06508.BD and UIDB/00285/2020].

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.jmsy.2023.03.001.

References

[1] Coronado E, Kiyokawa T, Ricardez GAG, Ramirez-Alpizar IG, Venture G, Yamanobe N. Evaluating quality in human-robot interaction: A systematic search and classification of performance and human-centered factors, measures and metrics towards an industry 5.0. J Manuf Syst 2022;63:392–410. http://dx.doi.org/10.1016/j.jmsy.2022.04.007.

[2] Li S, Wang R, Zheng P, Wang L. Towards proactive human–robot collaboration: A foreseeable cognitive manufacturing paradigm. J Manuf Syst 2021;60(July):547–52. http://dx.doi.org/10.1016/j.jmsy.2021.07.017.

[3] Marvel JA, Bagchi S, Zimmerman M, Antonishek B. Towards effective interface designs for collaborative HRI in manufacturing. ACM Trans Hum-Robot Interact 2020;9(4):1–55. http://dx.doi.org/10.1145/3385009.

[4] Zhang J, Wang P, Gao RX. Hybrid machine learning for human action recognition and prediction in assembly. Robot Comput-Integr Manuf 2021;72:102184. http://dx.doi.org/10.1016/j.rcim.2021.102184, URL https://www.sciencedirect.com/science/article/pii/S0736584521000673.

[5] Lucci N, Monguzzi A, Zanchettin AM, Rocco P. Workflow modelling for human–robot collaborative assembly operations. Robot Comput-Integr Manuf 2022;78:102384. http://dx.doi.org/10.1016/j.rcim.2022.102384, URL https://www.sciencedirect.com/science/article/pii/S0736584522000710.

[6] Chen C, Wang T, Li D, Hong J. Repetitive assembly action recognition based on object detection and pose estimation. J Manuf Syst 2020;55:325–33. http://dx.doi.org/10.1016/j.jmsy.2020.04.018, URL https://www.sciencedirect.com/science/article/pii/S0278612520300625.

[7] Xiong Q, Zhang J, Wang P, Liu D, Gao RX. Transferable two-stream convolutional neural network for human action recognition. J Manuf Syst 2020;56:605–14. http://dx.doi.org/10.1016/j.jmsy.2020.04.007, URL https://www.sciencedirect.com/science/article/pii/S0278612520300510.

[8] Brandli C, Berner R, Minhao Y, Shih-Chii L, Delbruck T. A 240 × 180 130 dB 3 µs latency global shutter spatiotemporal vision sensor. IEEE J Solid-State Circuits 2014;49(10):2333–41. http://dx.doi.org/10.1109/JSSC.2014.2342715.

[9] Duarte L, Neto P. Dataset of manufacturing tasks - DMT22. Dataset Zenodo 2023. http://dx.doi.org/10.5281/zenodo.7625962.

[10] Lichtsteiner P, Posch C, Delbruck T. A 128 × 128 120 dB 15 µs latency asynchronous temporal contrast vision sensor. IEEE J Solid-State Circuits 2008;43(2):566–76. http://dx.doi.org/10.1109/JSSC.2007.914337.

[11] Liu H, Brandli C, Li C, Liu S-C, Delbruck T. Design of a spatiotemporal correlation filter for event-based sensors. In: 2015 IEEE int. symp. circuits syst.. IEEE; 2015, p. 722–5. http://dx.doi.org/10.1109/ISCAS.2015.7168735.

[12] Delbruck T. Frame-free dynamic digital vision. In: Int. symp. secur. electron.. 2008, p. 21–6. http://dx.doi.org/10.5167/uzh-17620.

[13] Khodamoradi A, Kastner R. O(N)-space spatiotemporal filter for reducing noise in neuromorphic vision sensors. IEEE Trans Emerg Top Comput 2021;9(1):15–23. http://dx.doi.org/10.1109/TETC.2017.2788865.

[14] Guo S, Kang Z, Wang L, Zhang L, Chen X, Li S, Xu W. HashHeat: A hashing-based spatiotemporal filter for dynamic vision sensor. Integration 2021;81:99–107. http://dx.doi.org/10.1016/j.vlsi.2021.04.006.

[15] Lungu I-A, Corradi F, Delbruck T. Live demonstration: Convolutional neural network driven by dynamic vision sensor playing roShambo. In: 2017 IEEE int. symp. circuits syst.. IEEE; 2017, http://dx.doi.org/10.1109/ISCAS.2017.8050403.

[16] Wang Z, Ng Y, van Goor P, Mahony R. Event camera calibration of per-pixel biased contrast threshold. In: Australas. conf. robot. autom. ACRA. 2019, arXiv:2012.09378.

[17] Gallego G, Delbruck T, Orchard G, Bartolozzi C, Taba B, Censi A, Leutenegger S, Davison AJ, Conradt J, Daniilidis K, Scaramuzza D. Event-based vision: A survey. IEEE Trans Pattern Anal Mach Intell 2022;44(1):154–80. http://dx.doi.org/10.1109/TPAMI.2020.3008413, arXiv:1904.08405.

[18] Borer D, Delbruck T, Rösgen T. Three-dimensional particle tracking velocimetry using dynamic vision sensors. Exp Fluids 2017;58. http://dx.doi.org/10.1007/s00348-017-2452-5.

[19] Rueckauer B, Delbruck T. Evaluation of event-based algorithms for optical flow with ground-truth from inertial measurement sensor. Front Neurosci 2016;10. http://dx.doi.org/10.3389/fnins.2016.00176.

[20] Censi A, Strubel J, Brandli C, Delbruck T, Scaramuzza D. Low-latency localization by active LED markers tracking using a dynamic vision sensor. In: 2013 IEEE/RSJ int. conf. intell. robot. syst.. IEEE; 2013, p. 891–8. http://dx.doi.org/10.1109/IROS.2013.6696456.

[21] Tan JCA, Chan WP, Robinson NL, Croft EA, Kulic D. A proposed set of communicative gestures for human robot interaction and an RGB image-based gesture recognizer implemented in ROS. 2021, arXiv:2109.09908.

[22] Sheikholeslami S, Jung Moon A, Croft EA. Cooperative gestures for industry: Exploring the efficacy of robot hand configurations in expression of instructional gestures for human-robot interaction. Int J Robot Res 2017;36:699–720. http://dx.doi.org/10.1177/0278364917709941.

[23] Dreher CR, Wächter M, Asfour T. Learning object-action relations from bimanual human demonstration using graph networks. IEEE Robot Autom Lett 2020;5(1):187–94. http://dx.doi.org/10.1109/LRA.2019.2949221, arXiv:1908.08391.

[24] Neto P, Simão M, Mendes N, Safeea M. Gesture-based human-robot interaction for human assistance in manufacturing. Int J Adv Manuf Technol 2019;101:119–35. http://dx.doi.org/10.1007/s00170-018-2788-x.

[25] Wu C, Zhang J, Sener O, Selman B, Savarese S, Saxena A. Watch-n-patch: Unsupervised learning of actions and relations. IEEE Trans Pattern Anal Mach Intell 2018;40(2):467–81. http://dx.doi.org/10.1109/TPAMI.2017.2679054, arXiv:1603.03541.

[26] Zhang S, Huang H, Huang D, Yao L, Wei J, Fan Q. Subtask-learning based for robot self-assembly in flexible collaborative assembly in manufacturing. Int J Adv Manuf Technol 2022;120:6807–19. http://dx.doi.org/10.1007/s00170-022-09177-1.

[27] Heuss L, Gonnermann C, Reinhart G. An extendable framework for intelligent and easily configurable skills-based industrial robot applications. Int J Adv Manuf Technol 2022;120:6269–85. http://dx.doi.org/10.1007/s00170-022-09071-w.

[28] Amir A, Taba B, Berg D, Melano T, McKinstry J, Di Nolfo C, Nayak T, Andreopoulos A, Garreau G, Mendoza M, Kusnitz J, Debole M, Esser S, Delbruck T, Flickner M, Modha D. A low power, fully event-based gesture recognition system. In: 2017 IEEE conf. comput. vis. pattern recognit.. IEEE; 2017, p. 7388–97. http://dx.doi.org/10.1109/CVPR.2017.781.

[29] Baby SA, Vinod B, Chinni C, Mitra K. Dynamic vision sensors for human activity recognition. In: 2017 4th IAPR Asian conf. pattern recognit.. IEEE; 2017, p. 316–21. http://dx.doi.org/10.1109/ACPR.2017.136, arXiv:1803.04667.

[30] Ceolini E, Frenkel C, Shrestha SB, Taverni G, Khacef L, Payvand M, Donati E. Hand-gesture recognition based on EMG and event-based camera sensor fusion: A benchmark in neuromorphic computing. Front Neurosci 2020;14:637. http://dx.doi.org/10.3389/fnins.2020.00637.

[31] Innocenti SU, Becattini F, Pernici F, Del Bimbo A. Temporal binary representation for event-based action recognition. In: 2020 25th int. conf. pattern recognit.. IEEE; 2021, p. 10426–32. http://dx.doi.org/10.1109/ICPR48806.2021.9412991, arXiv:2010.08946.

[32] Duarte L, Safeea M, Neto P. Event-based tracking of human hands. Sens Rev 2021;41(4):382–9. http://dx.doi.org/10.1108/SR-03-2021-0095.

[33] Duarte L, Safeea M, Neto P. Event data for hand tracking - EDHT21. Dataset Zenodo 2021. http://dx.doi.org/10.5281/zenodo.4918320.