POLITECNICO DI TORINO
Repository ISTITUZIONALE

Online Classification of Human Gestures Through Event Camera Data Using a 3DCNN

Original
Online Classification of Human Gestures Through Event Camera Data Using a 3DCNN / Vico, L., Polito, M., Duarte, L., Pastorelli, S., Gastaldi, L., Neto, P.. - (2025), pp. 52-57. (2025 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC) Funchal, Madeira (PT) 02-03 April 2025) [10.1109/icarsc65809.2025.10970178].

Availability:
This version is available at: 11583/2999570 since: 2025-04-28T08:59:36Z

Publisher:
IEEE

Published
DOI:10.1109/icarsc65809.2025.10970178

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2025 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

19 November 2026

Online classification of human gestures through event camera data using a 3DCNN

Livia Vico
Dept. of Mechanical and Aerospace Engineering
Politecnico di Torino
Torino, Italy
livia.vico@studenti.polito.it

Michele Polito
Dept. of Mechanical and Aerospace Engineering
Politecnico di Torino
Torino, Italy
michele.polito@polito.it

Laura Duarte
Dept. of Mechanical Engineering
University of Coimbra
Coimbra, Portugal
laura.duarte@dem.uc.pt

Stefano Pastorelli
Dept. of Mechanical and Aerospace Engineering
Politecnico di Torino
Torino, Italy
stefano.pastorelli@polito.it

Laura Gastaldi
Dept. of Mechanical and Aerospace Engineering
Politecnico di Torino
Torino, Italy
laura.gastaldi@polito.it

Pedro Neto
Dept. of Mechanical Engineering
University of Coimbra
Coimbra, Portugal
pedro.neto@dem.uc.pt

***Abstract*—In environments where humans and robots interact and collaborate, the robot needs awareness of human actions and intentions to guarantee operational effectiveness and agent safety. In this paper, an online human action recognition system from event camera data is presented. A 3D Convolutional Neural Network (3DCNN) is built to classify event-based videos of isolated primitive assembly actions (idle, pick, place, screw), and then leveraged into classifying videos of action sequences. The 3DCNN is integrated into a system capable of acquiring, processing, and classifying real-time event data of assembly tasks. The proposed online system classifies the streamed data every 200 ms without accumulating delay.**

***Index Terms*—Online Classification, Manufacturing, Event data, Deep neural networks, 3DCNN**

I. Introduction

Human-robot interaction and collaboration is widespread, with examples of applications such as elderly care, education, medicine, and manufacturing [1]. In the manufacturing landscape, collaborative robotic systems are often adopted to work alongside humans on object manipulation and assembly tasks. The robot must be aware of human actions to ensure natural human-robot collaboration and safety [2].

Sensors acquire data about their environment that classification algorithms can map into suitable robot commands. The most commonly deployed sensors for tracking human movement are the Magneto-Inertial Measurement Unit [3] and camera systems with [4] or without markers [5], the latter often preferred. However, within this category, the commonly used RGB camera is limited by its fixed frame rate, low dynamic range and motion blur. In comparison, an event camera dynamically samples light based on changes in the captured scene [6]. An event camera captures brightness changes, called 'events', asynchronously and at a faster rate than an RGB camera, making it easier to extract features and more suitable for tracking dynamic motion such as human movement [7].

Machine learning (ML) plays a crucial role in classifying data that represent human gestures. A core ML method is the Deep Neural Network (DNN), an artificial neural network with multiple hidden layers of neurons. DNNs are particularly effective because of their ability to learn and recognize complex patterns from sensor data. Various DNN architectures have been applied to gesture recognition from event data, such as Spiking Neural Networks [8] and Long-term Recurrent Convolutional Networks [9], however, the former needs specialized hardware to operate effectively, and the latter requires separate parameter tuning for each network. Therefore, the 3D Convolutional Neural Network (3DCNN) architecture [10], [11] is preferred since it demonstrates good performance in capturing spatiotemporal information and does not share the aforementioned limitations.

This study proposes a system for gesture recognition and classification of event data. A 3DCNN is built and trained to recognize common primitive manufacturing gestures captured using an event camera. The system is first tested on offline classification tasks and then extended to perform online classification. The main contributions of this study are illustrated in Fig. 1 and listed as follows:

- Implementation of a 3DCNN capable of classifying manufacturing primitives from event data with high accuracy.

- Classification of continuous action data with a 3DCNN trained on primitives, i.e. single action videos.
- Online implementation of the proposed 3DCNN classification system.

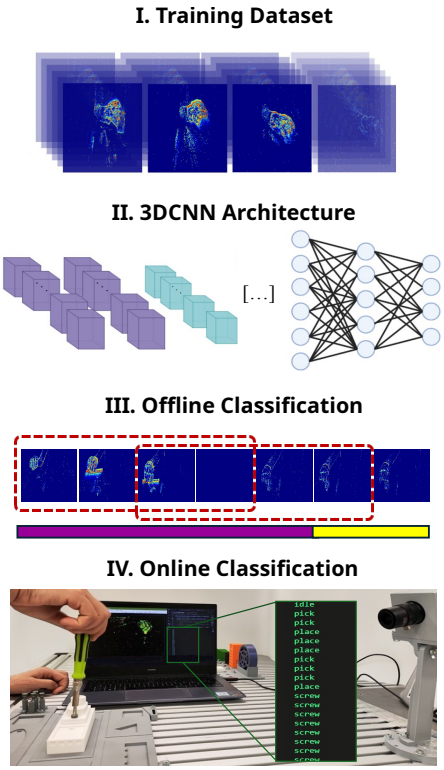


Fig. 1. Framework of the gesture recognition system.

II. System Setup

A. Event Camera

Inspired by biological vision systems, the event camera was created to respond to changes in brightness within a scene, with each pixel acting asynchronously. This approach samples light dynamically based on brightness changes in the captured scene. Each brightness change, called 'event' e, is encoded with the coordinates of the pixel detecting the change (x, y), a polarity pol specifying an increase or decrease in brightness, and a timestamp ts that indicates when the change occurred.

The Dynamic and Active Pixel Vision Sensor (DAVIS) is a widely adopted event camera model that captures greyscale images synchronously and event data asynchronously [12]. To extract meaningful information, event representations are often constructed from packets of events. A common example is to convert the events into an image, called an event frame, to be processed by image-based computer vision algorithms [13].

B. 3D Convolutional Neural Network

A 3DCNN mainly relies on convolutional and pooling layers. The 3D convolutional layers are the core of any 3DCNN, capturing spatial and temporal features within the data. The rectified linear unit (ReLU) activation function is applied on the result of the convolution to introduce non-linearity into data and allow the network to learn more complex dependencies within data. The 3D max pooling layers reduce dimensionality by downsampling the input into cuboidal regions and passing the maximum of each region to the next layer.

The input layer defines the fixed shape of the input tensor, which in the proposed network is set as $[n_{videos} \times n_{frames} \times 64 \times 64 \times 3]$. Note that the first two dimensions are variable, the first indicating the number of samples read by the network. The output layer corresponds to the number of categories into which the data can be classified, which in this case is 4 classes. It is characterized by a softmax activation function which normalizes the result of the previous layer into a probability distribution and outputs the class with the highest probability.

Other layer types are added to improve the robustness of the network. The use of dropout layers helps prevent overfitting, by randomly deactivating neurons during training. A flattening layer converts the multidimensional data into a single dimension. Fully connected layers allow the network to better learn combination of features rather than individual features. The 3DCNN proposed in this work, Fig. 2, has a structure based on a common 3DCNN architecture [14], and is implemented in Python using TensorFlow and Keras.

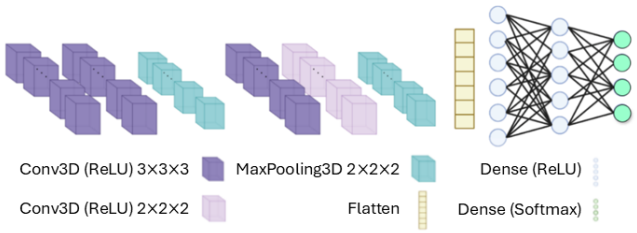


Fig. 2. Architecture of the proposed 3DCNN.

C. Event data processing

An event-based dataset of human gestures is required to provide data to train and test the 3DCNN. Although comprehensive datasets such as [15], [16] and [17] are often used for event-based gesture recognition, their emphasis is not on a manufacturing context. Therefore, the *Event-based Dataset of Assembly Tasks* (EDAT24) was selected, featuring 400 samples of four primitive manufacturing gestures: idle, pick, place, and screw [18]. The EDAT24 contains event data that must be processed to be a suitable input for the 3DCNN.

Since the required 3DCNN input is a sequence of 2D images, the time-based aggregation method called Temporal Binary Representation (TBR) is utilized to convert events into 2D images [10]. This method creates each event frame at a fixed time interval T by accumulating N intermediate binary representations. Each representation assesses the presence or absence of events for each pixel during $\Delta t = T/N$. The N temporally consecutive representations are aggregated in a tensor with a binary string of N bits and then converted into a decimal number for each pixel. Since this number is used to

set the pixel intensity for the resulting greyscale image, $N = 8$ is used to obtain the range of intensity values of $[0, 255]$, [10].

The TBR method can be adapted to work with fixed event accumulation or with an area accumulation strategy by changing the initial event accumulation condition for each frame. The resulting events per frame are then divided in the same way as in the original method, by creating N temporally consecutive representations during $\Delta t = T/N$. No significant differences in computational time and complexity were observed between the approaches. As such, the original TBR method is implemented to obtain predictions at a constant rate, with a value of $T = 100\,\text{ms}$ and thus $\Delta t = 12.5\,\text{ms}$.

Since the proposed 3DCNN requires an image input size of $64 \times 64 \times 3$, three operations must be performed on the generated TBR image to obtain a valid input for the 3DCNN, in the following order:

1) The greyscale TBR image is converted into a colored image, with 3 color channels, by applying the *Jet* colormap.
2) The resulting image is resized to $64 \times 64 \times 3$.
3) The pixel values are normalized to $[0, 1]$ by dividing each pixel by the highest brightness value in the image.

III. Offline Classification

A. Hyperparameter tuning

The network hyperparameters are iteratively redefined according to the results, until the best configuration is found. The values for dropout, batch size, and number of epochs that showed the best error curve and the highest accuracy were selected for subsequent evaluation steps, Table I.

TABLE I
HYPER-PARAMETER VALUES

Batch Size	Dropout	Number of epochs
16	0.4	25

Several tests were conducted to determine the minimum number of frames needed to guarantee accurate classifications. The results indicate that the accuracy increases with the number of frames in the window, reaching over 80% accuracy with 15 frames, as shown in Fig. 3. Furthermore, it was observed that the shortest sequence in the dataset takes 1.3 seconds, which corresponds to 13 frames. For this reason, a window size of 15 frames was chosen as a compromise between accuracy and window length.

Another parameter to consider is the overlap, as it determines the amount of new information introduced into each section for classification. If a sliding window moves with a 50% overlap, the current window shares half of its frames with the previous one, Fig. 4. It is crucial to choose the amount of frame overlap to avoid missing rapid movements while also avoiding highly redundant information. For this reason, several classifications with increasing overlap were performed to observe how accuracy varies. In an offline classification context, it is suitable to use the maximum overlap. At most,

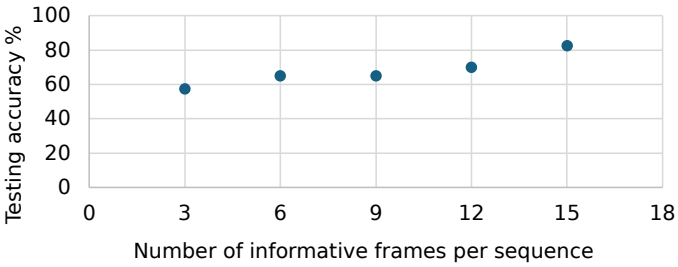


Fig. 3. Trade-off between classification accuracy and the number of frames required for each classification.

in this setup, the 15-frame windows can differ by a single frame, step size equal to 1, with a result of approximately 93% overlap.

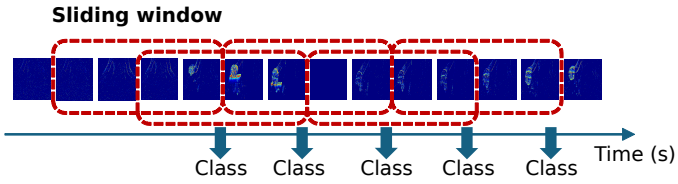


Fig. 4. Illustration of a 50% window overlap for a 4-frame window.

B. Training

Network training was conducted on the EDAT24 dataset. The videos of individual actions were segmented into 15-frame windows, with a 93% overlap between consecutive windows. An equal number of windows were selected for each class to ensure a balanced dataset. Specifically, a random subset was selected for the classes with a higher number of windows. Of each class in this dataset, 545 windows were used for training, while 61 were reserved for testing. The classification accuracy after training reaches 100% on the testing dataset.

C. Offline Testing

Once the trained network was built, it was tested on 9 continuous assembly sequences, some examples are reported in Fig. 5. All assemblies were performed by the same operator. The primitive actions occur consecutively rather than as isolated instances like the EDAT24 training data. Consequently, transitional movements between actions, which are difficult to classify, also occur within the videos. Additionally, while the order of actions is predetermined, the operator's speed is not constrained, resulting in no precise separation between actions.

Each window was labelled with one of the 4 possible primitive actions: idle, pick, place, or screw. The labels were assigned using the class of the most recent frame in the window and were subsequently visually inspected to verify and correct them, particularly in the transition movements between actions. To evaluate network performance, classification results were compared with ground truth through visual inspection. This approach is preferred over a confusion matrix or an F1 score analysis to better identify where and why the network struggles with multiclass classification in a continuous context.

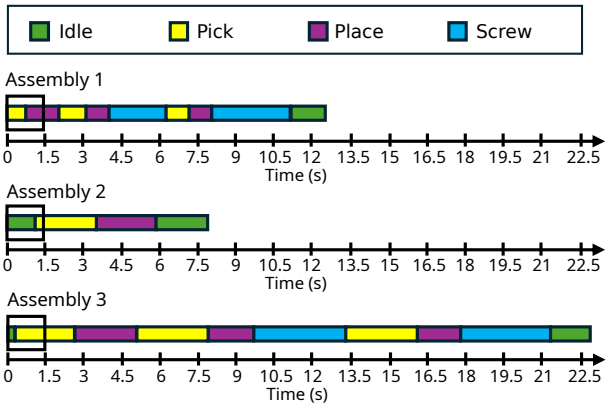


Fig. 5. Three continuous assembly sequences used to evaluate classification performance of the 3DCNN.

D. Results

The network performance is illustrated as the comparison between the classification result and the ground truth labels. This takes into account the continuous operation of the system, highlighting if delays occur and if the overall assembly is correctly reconstructed. Three different sequences of action primitives are analysed. For each window a prediction is made and reported in Fig. 6 at the starting time of the window.

The graph shows that the network is able to reasonably reconstruct the intended sequence of the assembly action. The screw class is consistently correctly classified since it has a recognizable movement pattern and the sequence always lasts longer than 1.5 seconds. The pick and place classes are also recognized in most instances. Most misclassifications occur in windows that contain multiple classes because the network is unable to prioritize one class. The transitional movements are another reason for frequent misclassifications between actions, because the network was not trained with these movements and does not know how to classify them. Moreover, at the end of each screw action, there is always a misclassification of place. This is because the movement of withdrawing the hand after releasing a screw is interpretable as the release of the object at the end of a place. The idle class in particular is rarely recognized because when the window contains movement, the network prioritizes it over the empty frames.

IV. Online Classification

A. System Implementation

The computer specifications used to run the tests are reported in Table II.

TABLE II
PC Specifications

Central Processing Unit (CPU)	AMD Ryzen 5 3500U (2.10 GHz)
Graphics Processing Unit (GPU)	Radeon Vega Mobile Gfx
Operating System	Windows 11

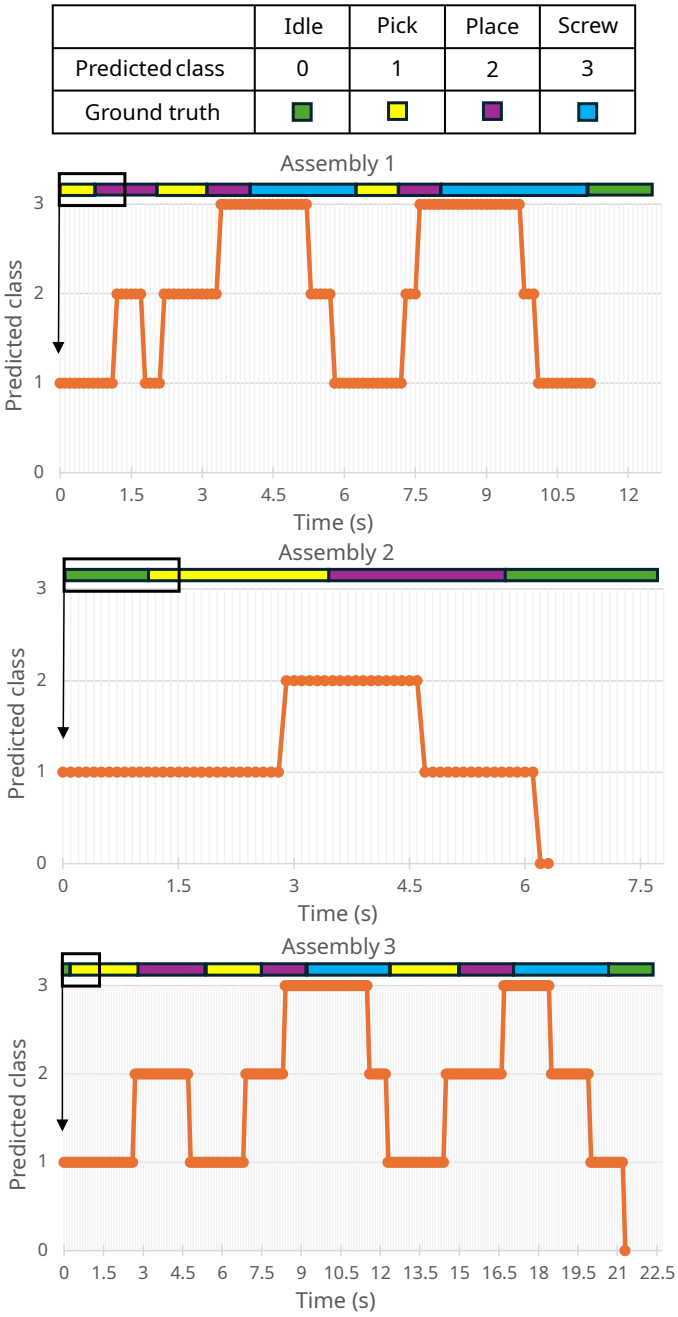


Fig. 6. Offline classification results for the continuous assembly sequences, the ground truth is reported on top of each plot.

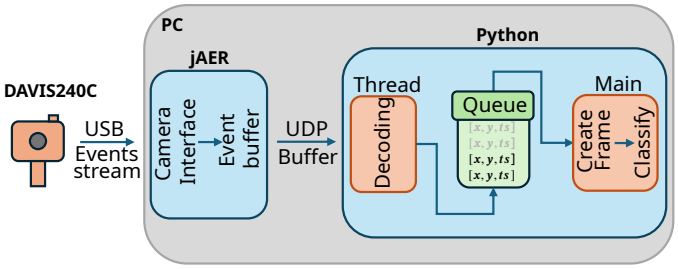


Fig. 7. System implementation logic, box with same colour represent program in parallel.

The classification system consists of different parts, including event acquisition, frame generation, and classification, as shown in Fig. 7.

The event acquisition is managed by jAER [19], which receives events from the event camera through a USB connection, stores the events in a buffer and then sends the buffer through UDP to the frame generation step. The UDP buffer size must be optimized. A decrease in buffer size leads to an increase in memory efficiency and reactivity but a decrease in computation efficiency. To address this trade-off, a buffer size of 1000 events was selected as a balanced compromise. Each event in the buffer is encoded with information about the pixel coordinates x and y and the event timestamp ts.

A dedicated Python thread is responsible for receiving the buffers, decoding each event pixel coordinates and timestamp, and constructing a list from this information $[x, y, ts]$. This list is appended to the end of a dynamic data structure called a queue according to FIFO (First In, First Out) logic. The queue ensures that the acquisition and classification processes can benefit from parallel processing.

When a new buffer of events is received, the main program calculates the cumulative duration of the unprocessed data within the queue. If this duration matches the predefined accumulation time of $T = 100\,\mathrm{ms}$, a frame is generated and added to the end of a group of 15 frames. The number of buffers required to generate a frame varies significantly based on the frequency of event generation at the given time, as high event activity accelerates buffer filling. A classification occurs when the 15-frame group has the right amount of new frames according to the selected overlap, substituting the equivalent number of oldest frames in the group. The same 3DCNN model used previously for offline classification is employed here.

B. Results

The described system was analysed in terms of temporal performance, by measuring execution times for each section of the system:

- **Decoding**: Time to decode the buffer received from the UDP and add the resulting list to the queue.
- **Read queue**: Time to retrieve a buffer from the queue until (but not including) TBR frame creation.
- **Create frame**: Time to create a TBR event frame.
- **Frame processing**: Time to colour, resize and normalize the new frame.
- **Classification**: Time to classify a group of 15 frames.
- **Cycle Time**: Time from the end of the previous processing cycle to the end of the current cycle, whether or not classification is included in the current cycle.

Within a cycle, the three longest processes are decoding the event data, creating a frame, and classifying the frame, Fig. 8. The classification time takes up to seven times longer than creating a TBR frame, the process with the second longest duration. Thus, the bottleneck of the system is the classification step. To ensure that the system works effectively, the classification overlap was reduced to 87%. This approach helps compensate for the delay caused by the classification while maintaining a high level of classification accuracy. As shown in Fig. 9 (Top), for an overlap of 93%, the system is not able to process all the data synchronously and leads to an increase in queue length.

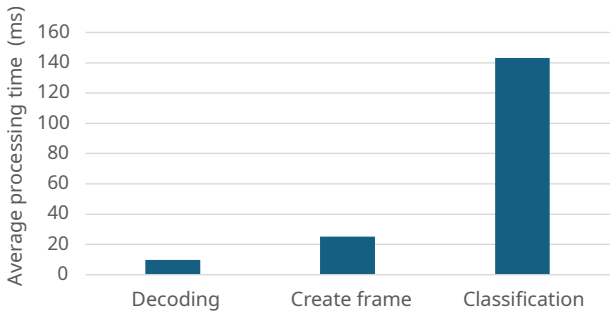


Fig. 8. Processing time required for the three longest process in the online classification system.

The time duration and the queue length for the online classification with 13 frames of overlap (87%), are reported in Fig. 9. As the queue length remains stable, Fig. 9 (Top), it is possible to conclude that the system is able to continuously process the information it receives. This is further supported by the results in Fig. 9 (Bottom), which show that the mean cycle time (i.e. the time needed to process new data) is shorter than the frame accumulation time (i.e. the time required to acquire new data). Moreover, analysis of the cycle time reveals two distinct cycle modalities: cycles in which classification occurs and cycles without classification. When no classification occurs, the cycle time closely matches the TBR. Since classification does not happen at every cycle and since all other processes require significantly less time than frame accumulation, the system achieves classification with only a small, consistent delay.

V. Discussion and Conclusion

This study proposes an online classification system capable of recognizing human gestures based on event camera data. The preliminary offline results prove that a system trained on action primitives is capable of classifying continuous sequences of action primitives. This generalization allows for a simpler data collection procedure, although it inherently struggles with transitions between primitives. A potential solution to this problem involves introducing an additional class to manage transitions or redefining the primitive gestures to incorporate transitions directly. An attempt was made to train the network with action primitives that did not include transitions. This led the network to classify them with existing classes such as pick and place. However, the classification accuracy remained mostly unchanged, indicating that the transition issue was not solved.

A stable queue length value indicates that data are processed as they are received, without accumulation of delay. The proposed system is capable of achieving this when the overlap is adjusted accordingly. The performance achieved can be

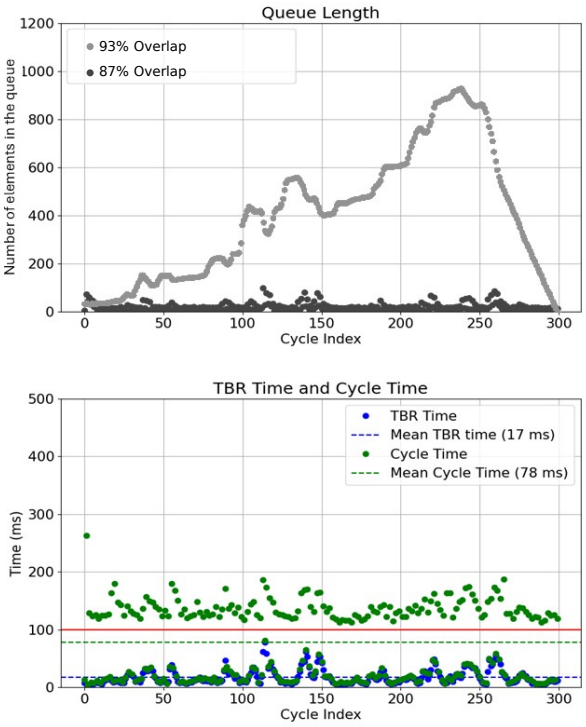


Fig. 9. Quantitative results from online classification system: Queue length for 87% and 93% overlap (Top); TBR and cycle time for 87% overlap, the red line represents the frame accumulation time. (Bottom).

considered adequate within the context of motion recognition. However, it could be further improved through the use of more powerful hardware, since the computer specifications play a crucial role in the computational time spent on each section of the system.

The next iteration of the proposed online classification system is the implementation of a predictive mechanism to identify the next task based on recognized actions. The proposed system can be utilized in an assembly sequence, where recognition plays a crucial role in identifying the current task. This information could enable the system to provide instructions to a robotic assistant, facilitating the next assembly step. For instance, the robot may supply the appropriate tool or component, or adjust the position and orientation of a part in accordance with the task. However, its deployment comes with some challenges, the most significant being misclassifications. They can not be totally removed, so a validation and correction logic should be implemented to ensure safety.

ACKNOWLEDGMENT

This research is sponsored by national funds through FCT – Fundação para a Ciência e a Tecnologia, under projects UID/00285 - Centre for Mechanical Engineering, Materials and Processes, LA/P/0112/2020 and 2021.06508.BD.

REFERENCES

[1] B. Chandrasekaran and J. M. Conrad, "Human-robot collaboration: A survey," SoutheastCon 2015, pp. 1–8, April 2015, DOI:10.1109/SECON.2015.7132964

[2] G. Hoffman and C. Breazeal, "Collaboration in Human-Robot Teams," Collection of Technical Papers - AIAA 1st Intelligent Systems Technical Conference, vol. 2, June 2012, DOI: 10.2514/6.2004-6434

[3] M. Polito, E. Digo, S. Pastorelli, L. Gastaldi, "Deep Learning Technique to Identify Abrupt Movements in Human-Robot Collaboration," in Proceedings of I4SDG Workshop 2023 (I4SDG 2023). Mechanisms and Machine Science, vol. 134, pp. 73—80. Springer, Cham. May 2023, DOI: 10.1007/978-3-031-32439-0_9

[4] R. Weitschat and J. Ehrensperger and M. Maier and H. Aschemann, "Safe and Efficient Human-Robot Collaboration Part I: Estimation of Human Arm Motions," in 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 1993–1999, May 2018, DOI: 10.1109/ICRA.2018.8461190

[5] Y. Wang, X. Ye, Y. Yang and W. Zhang, "Collision-free trajectory planning in human-robot interaction through hand movement prediction from vision," in 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids), pp. 305–310, January 2018, DOI: 10.1109/HUMANOIDS.2017.8246890

[6] G. Gallego et al., "Event-Based Vision: A Survey," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, n. 01, pp. 154–180, January 2022, DOI: 10.1109/TPAMI.2020.3008413

[7] C. Boretti, P. Bich, F. Pareschi, L. Prono, R. Rovatti and G. Setti, "PEDRo: an Event-based Dataset for Person Detection in Robotics," in 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Vancouver, BC, Canada, pp. 4065–4070, August 2023, DOI: 10.1109/CVPRW59228.2023.00426

[8] J. H. Lee et al., "Real-Time Gesture Interface Based on Event-Driven Processing From Stereo Silicon Retinas," in IEEE Transactions on Neural Networks and Learning Systems, vol. 25, no. 12, pp. 2250-2263, December 2014, DOI: 10.1109/TNNLS.2014.2308551

[9] J. Donahue et al., "Long-Term Recurrent Convolutional Networks for Visual Recognition and Description," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 4, pp. 677-691, April 2017, DOI: 10.1109/TPAMI.2016.2599174

[10] S. U. Innocenti, F. Becattini, F. Pernici and A. Del Bimbo, "Temporal Binary Representation for Event-Based Action Recognition," 2020 25th International Conference on Pattern Recognition (ICPR), Milan, Italy, pp. 10426–10432, May 2021, DOI: 10.1109/ICPR48806.2021.9412991

[11] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree and J. Kautz, "Online Detection and Classification of Dynamic Hand Gestures with Recurrent 3D Convolutional Neural Networks," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4207–4215, December 2016, DOI: 10.1109/CVPR.2016.456

[12] C. Brandli, R. Berner, M. Yang, S. -C. Liu and T. Delbruck, "A 240 × 180 130 dB 3 µs Latency Global Shutter Spatiotemporal Vision Sensor," in IEEE Journal of Solid-State Circuits, vol. 49, n. 10, pp. 2333–2341, October 2014, DOI: 10.1109/JSSC.2014.2342715

[13] Z. Jiang et al., "Mixed Frame-/Event-Driven Fast Pedestrian Detection," in 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, pp. 8332–8338, August 2019, DOI: 10.1109/ICRA.2019.8793924

[14] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun and M. Paluri, "A Closer Look at Spatiotemporal Convolutions for Action Recognition," in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 6450-6459, DOI: 10.1109/CVPR.2018.00675

[15] Y. Bi, A. Chadha, A. Abbas, E. Bourtsoulatze, and Y. Andreopoulos, "Graph-based Spatial-temporal Feature Learning for Neuromorphic Vision Sensing," in IEEE Transactions on Image Processing, vol. 29, pp. 9084—9098, September 2020, DOI: 10.1109/TIP.2020.3023597

[16] A. Amir et al., "A Low Power, Fully Event-Based Gesture Recognition System," in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7388—7397, July 2017, DOI: 10.1109/CVPR.2017.781

[17] Wang, Q. et al., "DailyDVS-200: A Comprehensive Benchmark Dataset for Event-Based Action Recognition," Computer Vision – ECCV 2024 (ECCV 2024). Lecture Notes in Computer Science, pp. 55-–72. Springer, Cham. October 2024, DOI: 10.1007/978-3-031-72907-2_4

[18] L. Duarte and P. Neto, "Event-based dataset for the detection and classification of manufacturing assembly tasks," Data in Brief, vol. 54, article 110340, June 2024, DOI: 10.1016/j.dib.2024.110340

[19] T. Delbruck, "Frame-free dynamic digital vision," in International Symposium on Secure-Life Electronics, University of Tokyo, pp. 21–26 March 2008, DOI: 10.5167/uzh-17620